MM/1

Technical

Manual

Written by: Mark Griffith and Carl Kreider
Publication date: October, 1991

Interactive Media Systems, Inc.
PO Box 1879
Davidson, NC 28036

Table of Contents

1 Recommended Software Practices

The following practices have been established to provide some level of standardization for software developers. The goal is to increase user convenience and consistency in the operation of MM/1 applications software as well as provide a common platform for development.

All developers are asked to please adhere to these standards.

1. All applications must restrain from direct manipulation of OSK memory variables except in extreme circumstances where no other method is possible. In those cases, written descriptions of what memory areas are changed and what values are inserted must be provided in the user documentation of the product. Further, direct writing to video memory will not be supported.

2. Use of system resources on a multi-user/multi-tasking computer must be as equally shared as possible to allow many processes to run concurrently. To do this, developers must not "hog" system resources by writing code such as "busy loops", using large amounts of memory when not needed, and so on. To achieve the goal of a "system friendly" program, developers are encouraged to use signals and events for I/O tasks, especially those requiring input from the keyboard or mouse. Also, developers must allocate to their applications large amounts of memory only when needed, and must return this memory to the shared pool when it is no longer needed.

3. Since the MM/1 has a unique console display system (graphics windows) and since all other OSK computer systems use a different type of display, developers are encouraged to write into their applications a test for the type of display and then act according to the response received. If the display is an MM/1 type window, full graphic features can be used. If it is not, then it would be wise to assume the display is an ASCII terminal. While this practice will make all applications increase in size, portability across the many different types of OSK system will greatly enhance the marketability of the product. Of course, some applications can only be accomplished on a graphic screen.

End of Section

2 System Overview

The MM/1 computer from Interactive Media Systems, Inc. is a low cost personal computer designed as an upgrade for experienced and new OS-9 users alike. The MM/1 provides the excellent programming and applications platform of the Microware OS-9/68000™ (OSK) operating system and combines with it the capabilities of a high-resolution graphics terminal.

The MM/1 comes bundled with the complete OS-9/68000 operating system including the "C" compiler, assembler, text editing and document processing tools, a terminal program, and many more software products.

2.1 Base System

The base system includes the processor board installed in a compact case with power supply and one 3.5 inch floppy disk drive. The main processor board consists of the following:

- Phillips SCC68070 CPU
- Phillips SCC66470 Video Controller
- 1 Megabyte of on-board memory
- Palette controller providing 16.7 million colors
- DB-9 analog RGB video port (15.7 Khz scan)
- 512K of ROM
- 1 DB-9 serial port
- 1 External serial port connection
- 34 pin dual header floppy disk interface
- 64 pin mini-bus for power and expansion board
- Interface for IBM style keyboards
- Inter-IC bus interface

2.2 Expansion Board

An optional expansion input/output circuit board is available. With the I/O expansion card, the following features are added:

- Sockets for 2 Megabytes of expansion RAM (SIMMs)
- 1 additional DB-9 serial port

- 2 additional external serial port connectors
- 2 bidirectional parallel port connectors
- 50 pin SCSI interface
- Battery-backed Real Time Clock

2.3 Floppy Drive Characteristics

The included 3.5 inch floppy disk drive has the following characteristics:

- Double or Single Sided 3.5 inch diskettes
- Variable sector size (default = 256 bytes)
- Maximum disk capacity 1.44 Megabytes

2.4 Additional Items

To use the MM/1 personal computer, the following additional items must be connected to the main processor board:

- Analog RGB monitor capable of syncing at 15.7 khz
- IBM style keyboard (88 or 101 keys) set for XT mode

2.5 Major Components

The following sections (2.5.1 through 2.5.9) provide an overview of the major components of the MM/1. See the references section at the end of this manual for more detail.

2.5.1 68070 CPU

The Phillips SCC68070 highly integrated microprocessor is a 16/32 bit Motorola 68000 compatible device designed to work in conjunction with the VSC video controller. The 68070 includes the following features:

- CHMOS technology
- 15 Mhz clock
- 32 bit internal data structures
- Two channel DMA controller
- On-chip UART serial interface
- Inter-IC serial bus interface
- 16 MByte addressing range
- 68000 object code compatibility

The on-board DMA controller is used for peripheral device to memory transfers, such as reading data from the floppy or hard disk drive. The maximum transfer rate is 3.2 Mbytes per second.

The on-chip UART serial interface supports asynchronous full or half duplex communications at speeds from 75 baud to 19.2 kbaud. Modem control is achieved by manipulating the CTS and RTS control lines. All features of this device are fully programmable.

The Inter-IC bus is a special synchronous serial bus designed for communications to other microprocessor and control devices. Up to 128 different devices can be on the bus at the same time. Maximum transmission speed is 100 kbits per second. The MM/1 can become either a Master or Slave device.

2.5.2 VSC Video Controller

The VSC device handles many system functions in partnership with the 68070 CPU. These devices were designed to complement each other. The VSC provides the following system functions:

- Initial system reset to all devices.
- Directly drives the on-board dynamic RAM.
- All control of the video display.

The VSC supports several types of displays and encoding methods. For example, the VSC can display bit-mapped, run length encoded, or mosiac encoded files. The following display types are used by the MM/1:

- 720 x 240 pixels in 16 colors
- 640 x 208 pixels in 16 colors
- 320 x 208 pixels in 256 colors

Using the interlaced mode doubles the vertical resolution to 416 lines for each display type, except for the 720 x 240 display.

2.5.3 Dynamic RAM

The 1 Megabyte of dynamic RAM on the MM/1 consists of eight 256 x 4 bit chips of 100 nanosecond or less access time. When the MM/1 is used as a 1 Megabyte system, this RAM serves as both video and system RAM. When used as a 3 Megabyte system, this RAM is used only as video RAM.

2.5.4 Floppy Disk Controller

The MM/1 includes a Western Digital WD37C65 floppy disk controller for control of up to 4 floppy disks. This device incorporates many functions that required external devices in older designs such as data separation, data rate selection, and clock generation. The interface circuits are designed to work with many common IBM-AT style floppy disk drives and can handle data rates from 125 kbits per second to 500 kbps. Up to four floppy drives can be connected at one time using a single 34 pin cable.

Currently, the WD37C65 controller used by the MM/1 will support the disk densities listed in Table 1.

Capacity	Type	RPM	Data Rate	Tracks	Secs/Trk
360Kb	5.25	300	250 Kb/s	40	9
1.2Mb	5.25	360	500 Kb/s	80	15
720Kb	3.5	300	250 Kb/s	80	9
1.44Mb	3.5	300	500 Kb/s	80	18

Table 1 Floppy Disk Types

These densities are what the chip is capable of handling. Different settings of the individual device descriptors will yield different disk densities than those listed above. See the section on Disk Drive Parameters (Section ???) for more information.

2.5.5 MC68901 Multi-Function Chip

The MC68901 multi-function peripheral chip provides a wide variety of functions and is designed to interface directly to a MC68000 microprocessor. In addition to its full-duplex UART, the MC68901 includes eight individually programmable I/O lines usually used as interrupt controls, as well as four individually programmable timers.

On the MM/1, two of these devices are used, one on the main processor board and the second on the addition I/O board. The processor board device provides serial port /t0 and the I/O board device serial port /t2.

2.5.6 SCSI Controller

The MM/1 additional I/O board includes a fully capable SCSI bus controlled by a Western Digital WD33C93A controller. This same device is used by IBM in many of their latest systems. This single chip implements all the required functions for full control of the SCSI bus and can transfer data at speeds of up to 4 Megabytes per second across the bus. As is usual for SCSI bus systems, up to 7 different devices can be on the SCSI bus at one time and under control by the WD33C93A.

In addition, the MM/1 was designed to make use of the new WD33C93B SCSI controller when it becomes available. This device provides all the functions as the A series and includes support for the new SCSI-2 standard, which includes the ability to transfer data across the SCSI bus at speeds of up to 10 Megabytes per second.

2.5.7 MC68681 DUART

The MC68681 is a dual asynchronous receiver/transmitter (DUART) that provides serial ports /t3 and /t4 on the MM/1 additional I/O board. This device is unique in its design and provides two serial ports from a single chip. In addition, the MC68681 has a maximum data transfer rate of up to 1 megabytes per second, full programmable control over all aspects of the I/O device port, ability to send and detect a BREAK, and many other features of interest to programmers and driver writers. Only one MC68681 is used on the MM/1.

2.5.8 MC68230 PI/T

The MC68230 is a dual parallel interface/time device. This chip provides the two parallel printer ports on the MM/1 additional I/O board. This device includes:

- Two bi-directional ports
- 8 or 16 bit capability
- One 24 bit timer

The bi-directional capability of these ports provide increased possibilities for unique applications for communications, industrial control, and many other applications where high-speed, 8 or 16 bit data transfers are required.

2.5.9 DS1287 Real-Time Clock

The Dallas Semiconductor DS1287 RTC is included on the additional I/O board to provide a real-time system clock for the MM/1. This device includes an internal battery rated for a 10 year life. Once set the first time (using the SETIME utility), this device will never need to be reset during the life of the battery.

End of Section

3 System Memory

The MM/1 has two basic memory maps -- one when booted as a 1 Megabyte system, and another when booted as a 3 Megabyte system. The size of the memory available for use is controlled by a jumper on the processor board at P7 (see Section ? for more details).

3.1 Memory Map

Table 2 below shows how the memory for both configurations is used.

3.2 Hardware Memory Management

The following sub-sections explain how the ROM, Video RAM, Process RAM and Expansion RAM are managed by OS-9/68000.

3.2.1 Read Only Memory (ROM)

The memory map in Table 2 above takes some explaining to see how it all works. The ROM actually appears at address _$180000 when power is first applied to the system. This is necessary because the VSC looks at that address for the first four instructions to get the system going.

After the VSC has read those four bytes, the ROMs are then re-mapped at _$980000 to move them out of the contiguous memory used by OS-9. You can see the ROM at this address on a 3 Megabyte system by looking at the output of the MDIR -E utility in Table 3.

On a 1 Megabyte system, the ROMs are mirrored at the _$180000 location.

3.2.2 Video Memory

The memory used by the VSC also does not appear at a location that one would expect. This memory is addressed at _$800000 to _$900000, but also appears to mirror itself at _$000000 to _$100000 in a 1 Megabyte system. This, of course, is necessary for the 1 Megabyte machine to be able to use that memory as both video and system/user RAM, since OS-9 must see RAM at _$000000 to load the operating system at boot-up.

HEX Address	Used as
$1000000 $9F0000	Used by hardware I/O devices
$9A0000 $980000	ROM
$900000 $800000	1 Megabyte of video RAM - both 1 Megabyte and extended memory systems
$7FFFFF $200000	Continuation of user memory when booted as a 9 Megabyte system
$19FFFF $17FFFF	ROM appears here on a 1 Megabyte system
$17FFFE $100000	Continuation of user memory when booted as a 3 Megabyte system
$0FFFFF $000000	System/user memory – also video memory mirrored on a 1 Megabyte system

Table 2 Memory Map

In a 3 megabyte system, the video memory appears at its normal location. This memory then becomes "colored" memory as far as OS-9 is concerned. Colored memory is basically memory that is reserved for a specific purpose, like video RAM, and will not be used by the operating system except as a last resort. This means that OS-9 will not touch the video RAM unless it (the operating system) becomes desperate for memory to fulfill a task. In a 1 Megabyte system, this would not happen since the video and system must share the same memory.

Addr	Size	Owner	Perm	Type	Revs	Ed #	Lnk	Module Name
00018d8c	418	0.1	0555	Prog	0000	5	1	sysgo
0098553c	18052	1.0	0555	Trap	c009	6	2	cio
00989bc0	20754	1.0	0555	Prog	c001	52	3	shell
0098ecd2	4702	1.0	0555	Prog	c001	24	0	setime
0098ff30	5214	1.0	0555	Prog	c001	20	2	mdir
0099138e	2736	1.0	0555	Prog	c001	15	0	mfree
00991e3e	5440	1.0	0555	Prog	c001	21	0	procs
0099337e	3146	1.0	0555	Prog	c001	4	0	devs
00993fc8	15598	0.0	0005	Prog	c002	33	0	format
00997cb6	9700	0.0	0555	Prog	c001	39	0	dir
0099a29a	7586	1.0	0555	Prog	c001	29	0	copy
0099c03c	2426	1.0	0555	Prog	c001	15	0	iniz
0099c9b6	5536	1.0	0555	Prog	c001	23	0	free
0099df56	3284	1.0	0555	Prog	c001	16	0	load
0099ec2a	2366	0.0	0555	Data	8080	1	1	stdfonts
0099f568	144	0.0	0555	Subr	8001	0	0	syscall
0099f5f8	2456	0.0	0555	Prog	c001	1	0	break
001fbaf0	10112	0.0	0777	Data	8080	1	3	WData

Table 3 ROM Directory

Thus, a 3 Megabyte system has 2 Megabyte of user RAM from address _$000000 to _$200000 and 1 Megabyte of video RAM at _$800000 to _$900000.

3.2.3 MFREE -E Output

As another example of how the memory is allocated, here is the output of the MFREE utility on a 3 Megabyte machine:

```
Minimum allocation size:        0.25 K-bytes
Number of memory segments:    11
Total RAM at startup:         2996.25 K-bytes
Current total free RAM:       1953.50 K-bytes

Free memory map:

Segment Address             Size of Segment
----------------    --------------------------
_$4D00            _$300              0.75 K-bytes
_$1AC00           _$200              0.50 K-bytes
_$1B000           _$B00              2.75 K-bytes
_$23E00           _$133D00        1231.25 K-bytes
_$181F00          _$300              0.75 K-bytes
_$1ED700          _$100              0.25 K-bytes
_$1EFD00          _$500              1.25 K-bytes
_$1F0500          _$100              0.25 K-bytes
_$1F1100          _$100              0.25 K-bytes
_$1F9800          _$300              0.75 K-bytes
_$801400          _$B2B00          714.75 K-bytes
```

Notice that the video memory begins at address _$801400. In my machine when this utility was run, only 3 windows were opened, giving me 714.75 Kbytes of available video memory. On a 1 Megabyte system, memory actually in use by the video controller (VSC) can be anywhere within the 1 Megabyte address range, depending upon when it was allocated. Refer to the section of memory fragmentation for more information.

For more information on colored memory, refer to the OS-9/68000 Technical Manual, page 2-11.

3.2.4 Process Memory Allocation

OS-9 allocates memory to processes at the top of memory, whether it is a 1 or 3 Megabyte system, and moves "downward" as more processes are loaded. This serves several purposes. Not only does this protect the OS-9 system variables that are at the extreme lower end of the memory map, but it also prevents a system using colored memory from allocating that "protected" memory until all the other available memory on the system has been used. In the case of the MM/1, the video memory

would then be the last memory used in a memory-starved machine. Of course, this is not true with a 1 Megabyte system, since system and video memory share the same physical space.

Notice in the MFREE output above that the largest block of unallocated memory is at _$23E00 and is _$133D00 bytes large. Notice also that the memory above this area is the most fragmented, since this is where all the memory modules have been loaded and run. As the system is used, more fragmentation will occur, the large block of unallocated memory will become smaller, and the amount of fragmentation in the upper ranges of memory will increase.

One caveat though -- once all the video memory is used for windows, OS-9 will not allocate more video memory from the user RAM areas, although the opposite is true as explained above.

3.2.5 Expansion RAM Access

As a side note, when the MM/1 is booted as a 1 Megabyte system, it is still possible to access the 2 or 8 megabyte of expansion RAM if it is installed in the I/O board. In this case, that memory appears as address _$400000 to _$600000 or _$800000 depending on the RAM size. This is useful for running memory tests or for just playing around with the extra RAM. There are no system services that will allow you to access that RAM, but it can be directly addressed.

3.3 Memory Fragmentation

On an OS-9 Level I or OS-9/68000 system, memory modules are loaded into memory as they are required, processes needing additional memory are given what is available at the time of the request. All this creates an addressable memory area that is broken up into little pieces, or fragmented. This discussion cannot cover the scope of this area, but is given only as a means for new users to better understand how the MM/1 and OS-9/68000 memory allocation works. An excellent discussion of OS-9 Level I and Level II memory fragmentation is given in The Complete Rainbow Guide to OS-9 by Dale Puckett and Peter Dibble, published by Falsoft Inc., Prospect Kentucky.

3.3.1 INIZing Devices

One method to use to insure memory fragmentation is kept to a minimum is to INIZ I/O devices as soon as the system as started. Doing this in the startup file is a good choice. What this does is to allocate the static memory used by I/O devices at the very top of user RAM, before any other modules are loaded. Once these static memory areas are allocated, that device won't need to use memory in any other

areas of RAM, thereby reducing one of the major causes of memory fragmentation. Using the DEVS utility, you can see where the memory for these INIZed devices is allocated:

```
IMS MM/1 68070 Computer 3 MB System OS-9/68K V2.4

Device      Driver      File Mgr    Data Ptr   Links
----------  ----------  ----------  ---------  ------
term        windio      scf         $001fe270      2
dd          rbsccs      rbf         $001facf0      9
h1          rbsccs      rbf         $001facf0      1
w1          windio      scf         $001f63f0      3
w2          windio      scf         $001f1500      3
p           sc68230     scf         $001f6300      1
nil         null        scf         $001f9640      1
```

INIZing a device after the system is running and other modules are loaded will not allocate memory from the top of RAM, but rather from the top of whatever RAM is available. Remember, since OS-9 gives memory to processes from the top of available RAM on down, the allocated memory for the INIZed device may fall in the middle of the memory map, thereby fragmenting it badly.

3.3.2 "Sticky" Modules

Another method OS-9/68000 uses that OS-9/6809 did not have to control memory fragmentation is "sticky" modules. These are modules that once loaded, they remain in memory even after the program has stopped execution. This cuts down on fragmentation by keeping the module around in memory under the assumption that most of the programs run early in the system's current life-span would be loaded in more-or-less contiguous memory blocks.

However, this is not true in everyday work. Modules remaining in memory can make fragmentation worse just by the fact that they "hang around" after they are finished. One way to manually control this is to get rid of these "klingon" (he he he) modules by zapping them into nonexistence with the UNLOAD utility. One caution here--make sure you LOAD the UNLOAD utility in the startup file. If not, the first time you run UNLOAD, it will stick in memory itself and cause even more fragmentation. Loading it first will at least make sure it is loaded in a high memory area and will always be there when you need it.

It is not normally a good idea to INIZ serial port devices. When this happens, DTR for that port is set and will remain so. If a path is then opened to that port with, for

example, a terminal program, communication will proceed normally until the operator terminates the program. At this point, since the port was INIZed, DTR will not drop and the modem may not hang up.

End of Section

4 Customizing the System

The OS-9/68000 operating system is based on the concept of modular construction. Each part of the operating system consists of a certain module that performs some sort of service, such as controlling a disk drive device, sending and receiving data from the serial ports, or controlling these operations. While the entire concept of how OS-9 is structured is beyond the scope of this manual, some knowledge is required to make a customized bootfile for your system.

4.1 Bootfiles

Once the MM/1 finishes its initialization after power up, it begins looking for a boot record on the default device, usually /d0. If this bootfile is found, the MM/1 loads it and begins execution.

The bootfile is simply all the different modules that are required to run the system merged into one file. This file is loaded at a specific address and the microprocessor jumps to the start of this code to begin running the operating system. What must be in the bootfile, and how to get it there is explained in the next few sections.

4.1.1 System Modules

The MM/1 and all other computers running the OS-9/68000 operating system require several modules to be in memory before the operating system can run. These modules, and their purposes, are explained below:

Kernel	The core of the system
Init	The system initialization module
tk68901	Clock driver module
Scf	Sequential character manager
windio	Windowing system manager/driver
Term	First display device
Rbf	Disk drive manager
d0	Default disk drive descriptor
Sysgo	System startup module

With this bootfile, the MM/1 will start and run the Shell. Very little else can be done at this point, so this exercise only serves to show the minimum modules that are

required to get the system running. Before you begin including more modules in your bootfile, you might need to know what modules serve what purpose.

4.1.2 More Memory Modules

Depending upon the needs of your system, you would include different modules in the bootfile. For example, if you do not need a Ramdisk, you would not include those modules. However, you might want to include modules for two serial ports (out of the five that are available), and only one of the two parallel port devices. In this way, you customize the system to your specific needs, and also save memory by including only those modules you need.

Although OS-9/68000 allows you to load modules after the system is booted and to use those modules as if they were included in the original bootfile, it is many times more desirable to include all the modules you might need in the bootfile. Doing this insures that all the modules are loaded into contiguous memory blocks thereby reducing memory fragmentation.

Sections 4.1.3 through 4.1.10 describe the modules included with the MM/1.

4.1.3 System modules

kernel	The OSK kernel
Init_base	The system initialization module
init_3meg	Init module for 3 Megabyte systems
sysgo	The module that starts the system
tk68901	Clock driver module
rtcds1287	Real Time Clock module

4.1.4 File managers

rbf	Random block file manager
scf	Sequential Character file manager
sbf	Sequential block file manager
pipeman	Pipe file manager
nfm	Network file manager

4.1.5 Device drivers

windio	The window device driver
rbvccs	SCSI device driver
scsi_mm1	Sub-module for the rbsccs module
rb33c93	SCSI device driver (older systems)
rb37c65	Floppy disk device driver

sc68070	Driver for serial port /t0
sc68230	Driver for the printer ports
sc68681	Driver for serial ports /t3 and /t4
sc68901	Driver for serial ports /t1 and /t2
ms901	The serial mouse driver
n9026	ArcNet device driver
n6850	Serial network device driver
null	Driver for nil device

4.1.6 RBF disk devices

d0	Floppy drive 0 descriptor
d1	Floppy drive 1 descriptor
d2	Floppy drive 2 descriptor
d3	Floppy drive 3 descriptor
st0	Descriptor for the Atari disk 0
st1	Descriptor for the Atari disk 1
dd.d0	D0 as the default device /dd
dd.h0	H0 as the default device
dd.st0	ST0 as the default device
h0	Hard drive descriptor

4.1.7 SCF Serial devices

t0	Device descriptor for serial port /t0
t1	Device descriptor for serial port /t1
t2	Device descriptor for serial port /t2
t3	Device descriptor for serial port /t3
t4	Device descriptor for serial port /t4
term	The console device descriptor
p	Parallel printer descriptor
p1	Second printer descriptor
ms	The serial mouse descriptor (/t2)

4.1.8 Network devices

n0	ArcNet device descriptor
ns2	Serial network device descriptor 2
ns3	Serial network device descriptor 3

4.1.9 Misc modules

nil	The "do nothing" device
pipe	Pipe device descriptor

r0	Ramdisk R0
ram	Ramdisk device driver

4.1.10 Window devices

w	The "generic" window device descriptor
w1	Numbered window device descriptors
.	/w1 through /w12
.	
w12	

4.1.11 Making Bootlists

To customize the modules in the bootfile, and therefore the modules that are loaded into memory at system startup, all a user needs is to make a list of those modules they would like to use, and run the OS9GEN utility. This list, called a 'bootlist', is simply a text file that lists all the modules in the order you what them to appear in the new bootfile. For example, here is a simple bootlist to make a bootfile for a 3 Megabyte system. Lets say it is called 'bootlist.3meg':

```
Kernel
Init_3meg.
tk68901
rtcds1287
Scf
sc68070
sc68230
sc68681
sc68901
windio
Term
w
w1
w2
T0
T1
T2
P
Null
Nil
Pipeman
Pipe
```

Rbf
rb37c65
rbsccs
scsi_mm1d
d0
st0
h0
h1
dd.h0d
ms901
ms
Sysgo

Note: If the modules are not in the current directory, then absolute paths to each module must be given in the bootlist file.

You make the bootlist by using any text editor, such as Umacs or microEmacs that are shipped with the MM/1.

4.1.12 Module Order

When constructing the bootlist file, there are a few considerations that must be followed:

1. The KERNEL file MUST be the first module in the bootlist, hence the first module loaded into memory when the system boots. This is critical.

2. Every device descriptor in the bootlist must have its corresponding driver included also. Consult the list in Sections 4.1.3 through 4.1.10 to see which driver goes with which module.

3. As a matter of style, it is also customary to list the various devices in the bootfile immediately below the driver and file manager for those devices, as in the listing of SCF and RBF drivers and descriptors above.

4.1.13 Using OS9Gen

Once the bootlist file has been made, the next step is to create the new bootfile by using the OS9Gen utility. By using the HELP utility, you can get an idea of the options to use with OS9Gen:

Syntax: os9gen {<opts>} <device> {<path>} {<opts>}
Function: create boot on disk
Options: -b=<size> copy buffer size (default 64k)
-e extended boot (large >64k or fragmented)
-x pathlists relative to execution directory
-z=<path> read list of files from standard path

NOTE: Not all options listed

To create the bootfile on drive /d0, you would then enter the command:

```
_$ os9gen -e -b=128k /d0 -z=bootlist.3meg
```

Normally, bootfiles cannot be larger than a 64K limit. The -e option tells OS9Gen to create an extended bootfile and the -b option tells it how big this file can be. Note that the actual bootfile may be smaller than the size given for the -b option, but it cannot be larger than this.

The -e option also tells OS9Gen that the bootfile need not be one contiguous. This is important when making a bootfile on a badly fragmented disk, or on a hard disk drive.

Once the new bootfile has been made, rebooting the machine with that boot disk in place will cause the new modules to be loaded as the system boots. If the memory modules included are still not as you need, the bootlist file can be edited and OS9Gen run again.

4.2 Using MODED

The MODED utility provided by Microware is a very useful tool to use when modifying system modules. In some circumstances, MODED is the only tool that can get the job done.

4.2.1 The INIT module

One case in particular is the INIT module. This module is used to startup the system and establish many system wide default parameters. To edit these parameters, the MODED utility is used. Many of the parameters you are allowed to change are best left alone. *Changing parameters without knowing exactly what you are doing can result in a machine that no longer works, or works incorrectly.*

There are, however, some parameters that the user will need to change depending upon the type of system they plan to run. For example, when you have a floppy-based system and are installing a hard disk drive. You would want to boot the system with the hard disk drive as the new default device of /h0. To do this you would change the default directory name using MODED from /d0 to /h0. You can also change the default console device from /term to an external terminal port if you like by again using MODED to change the default I/O path from /term to /t1 or /t3, for example.

Changing parameters within the INIT module without knowing exactly what you are doing can result in a system that will not run, or is not compatible with other OSK system. Please be careful.

Consult the Microware manual, page 2-15 for further examples of how to use MODED and the INIT module.

End of Section

5 How it Works

The heart of the MM/1 consists of the 68070 CPU and the SCC66470 video and system controller (VSC). The remainder of the components on the boards are actually peripheral devices the CPU can address through certain memory locations. Please refer to Figure 1 below

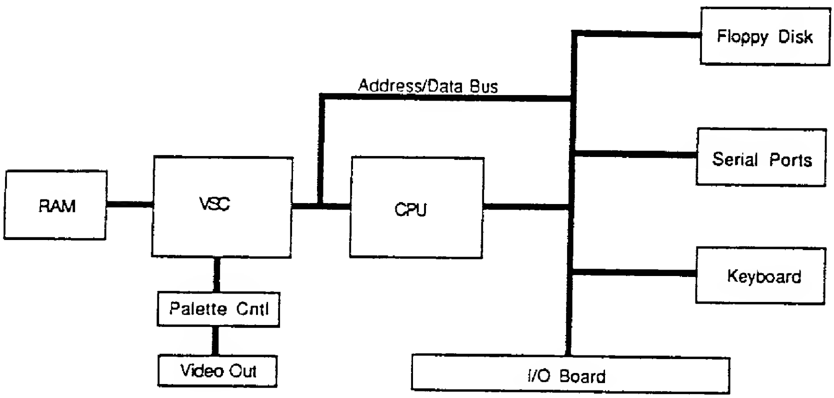


Figure 1 MM/1 Block Diagram

5.1 The VSC

The VSC (Video and System Controller) provides all the video processing for the MM/1 along with various other functions such as refresh for the 1 Megabyte of DRAM used for video and system RAM and bus arbitration and control along with the 68070 CPU. The video processing component of the VSC is designed to be compatible with European, Japanese, and USA standards for TV and video. The

VSC, working with the Brooktree palette controller, provides the analog RGB output of the MM/1.

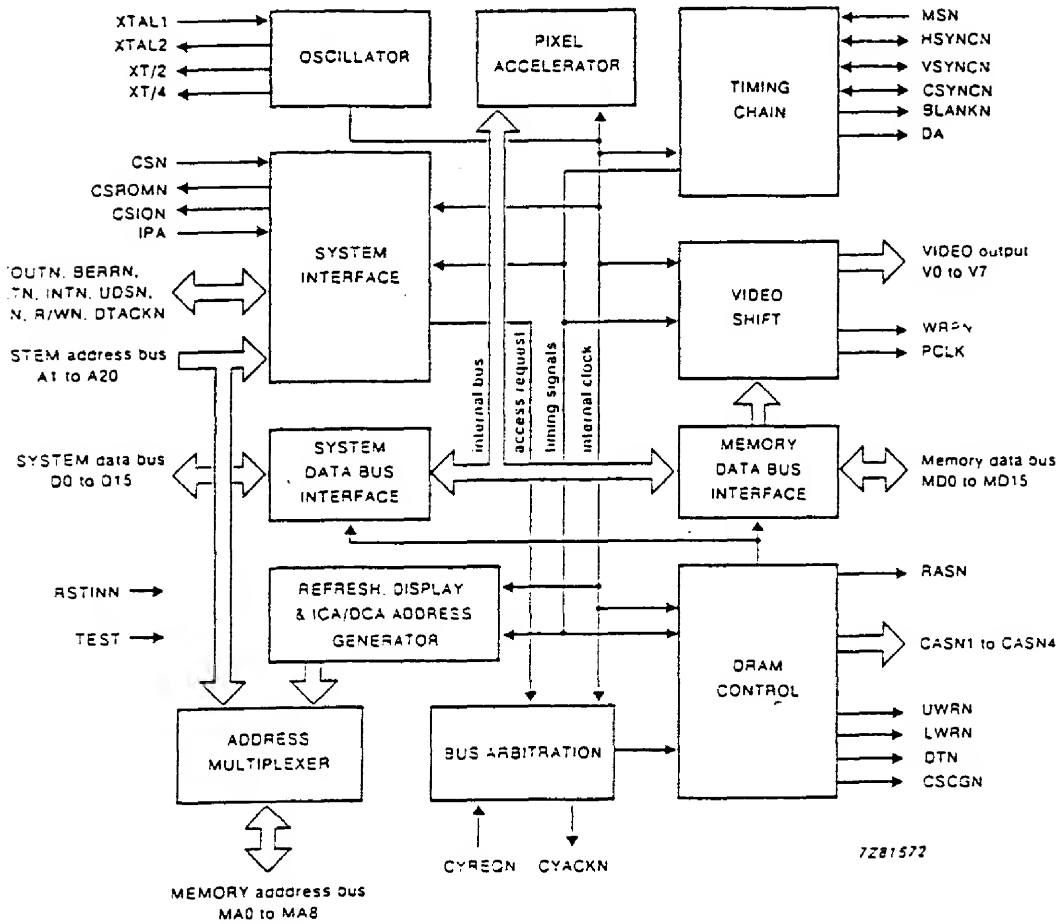


Figure 2 VSC Block Diagram

5.1.1 Video Memory Access

The 1 Megabyte of memory on the processor board is actually controlled by the VSC. Even the CPU must "go through" the VSC to access this RAM, which the VSC allows at certain points in the video cycle.

The reason 1 Megabyte systems are noticeably slower then those using the expansion RAM is because the CPU must wait for the VSC to finish its display mode before it has access to the RAM. Each video line that is displayed consists basically of a refresh period where the RAM is "recharged", the display period, and then a free period. It is during this period that the CPU can run freely.

The VSC, of course, provides all video controls. Horizontal and vertical sync signals are provided from the VSC directly to the video port. Video output is achieved by sending an 8-bit word corresponding to the contents of a video RAM memory location to the palette controller, which performs the conversion to analog RGB signals. Control of the many different display modes and the rest of the VSC functions are achieved by writing into control registers within the VSC. For more information on programming the VSC, consult the documents referred to in the reference section at the back of this manual.

5.2 The CPU

The SCC68070 CPU is more than just a microprocessor, it is a complete microcomputer system. By adding some RAM and ROM, and mating with the VSC, the 68070 can function almost entirely on its own. Consult Figure 3 below.

5.2.1 CPU Programming Model

The 68070 is functionally the same as a Motorola 68000 series microprocessor. Its instruction set is essentially the same and all programs written for a 68000 will run without change. See Figure 4 through Figure 6.
For more information concerning programming the 68070, consult the documents listed in the reference section at the end of this manual.

5.2.2 Interrupt Levels

The CPU controls the peripheral devices through reading and writing to certain memory locations. For example, to access data from the floppy disk drives, the operating system causes the CPU to send a string of control bytes to the floppy disk controller's Master Status Register. Once this is done, the operating system "goes to sleep" and waits for the floppy disk controller to "interrupt" it's sleep with a signal telling it a sector of disk data is ready to be read. The operating system then causes the CPU to read into memory the 256 data bytes the disk controller has pulled

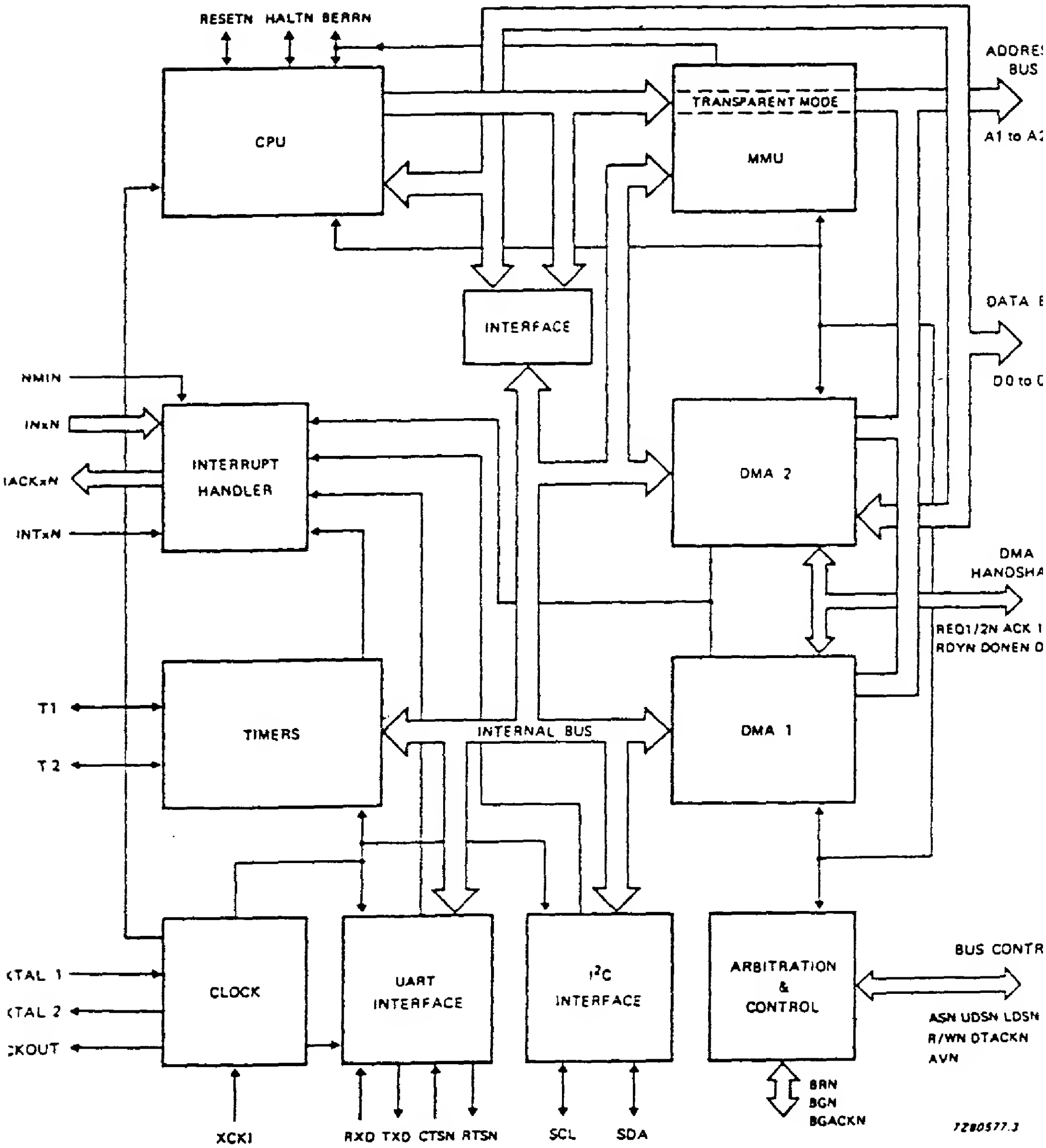
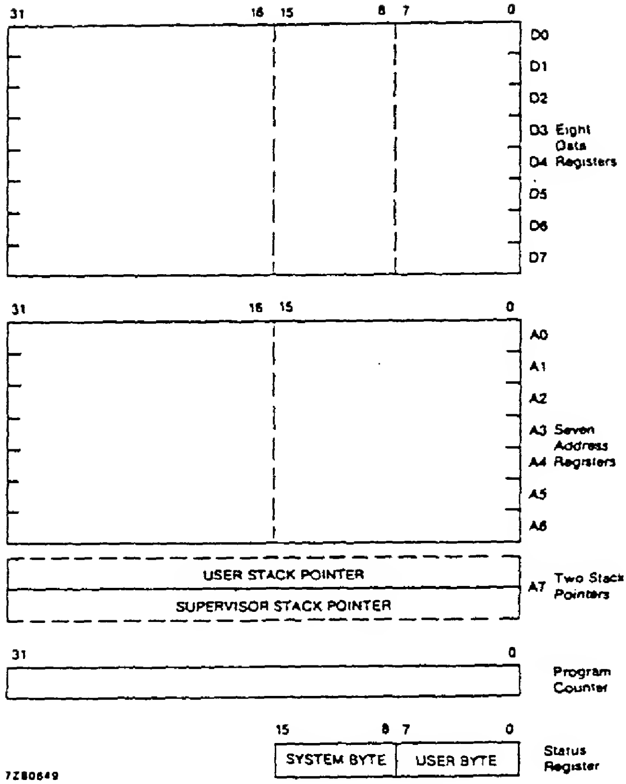
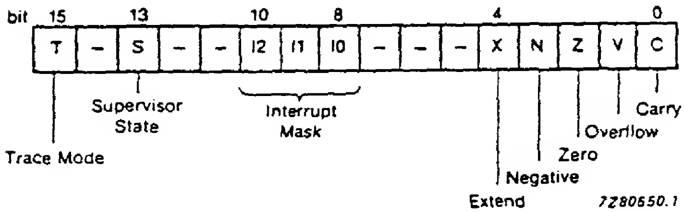


Figure 3 CPU Block Diagram



Programming model



Status register

Figure 4 Programming Model

Fig. 2.3a Bit data (1 byte = 8 bits)

bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

MSB BYTE 0 LSB	BYTE 1
BYTE 2	BYTE 3

7Z80652

Fig. 2.3b Integer data (1 byte = 8 bits)

bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

MSB WORD 0 LSB
WORD 1
WORD 2

7Z80653

Fig. 2.3c Word data (16 bits)

bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

MSB LONG WORD 0	HIGH ORDER
	LOW ORDER LSB
LONG WORD 1	HIGH ORDER
	LOW ORDER
LONG WORD 2	HIGH ORDER
	LOW ORDER

7Z80654

Fig. 2.3d Long-word data (32 bits)

bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

MSB ADDRESS 0	HIGH ORDER
	LOW ORDER LSB
ADDRESS 1	HIGH ORDER
	LOW ORDER
ADDRESS 2	HIGH ORDER
	LOW ORDER

7Z80655

Fig. 2.3f BCD data (2 BCD digits = 1 byte)

bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

MSD BCD 0	BCD 1 LSD	BCD 2	BCD 3
BCD 4	BCD 5	BCD 6	BCD 7

7Z80656

Fig. 2.3e Addresses (1 address = 32 bits)

Figure 5 Register Maps

Table 2.5 Instruction set

Mnemonic	Description	Mnemonic	Description
ABCD	Add Decimal with Extend	MOVE	Source to Destination
ADD	Add	MULS	Signed Multiply
AND	AND Logical	MULU	Unsigned Multiply
ASL	Arithmetic Shift Left	NBCD	Negate Decimal with Extend
ASR	Arithmetic Shift Right	NEG	Negate
Bcc*	Branch Conditionally	NOP	No Operation
BCHG	Test Bit and Change	NOT	Logical Complement
BCLR	Test Bit and Clear	OR	Inclusive OR
BRA	Branch Always	PEA	Push Effective Address
BSET	Test Bit and Set	RESET	Reset External Devices
BSR	Branch to Subroutine	ROL	Rotate Left without Extend
BTST	Test a Bit	ROR	Rotate (Without Extend)
CHK	Check Register against Bounds	ROXL	Rotate Left without Extend
CLR	Clear an Operand	ROXR	Rotate with Extend
CMP	Compare	RTE	Return from Exception
DBcc	Test Condition, Decrement & Branch	RTR	Return and Restore Condition Codes
DIVS	Signed Divide	RTS	Return from Subroutine
DIVU	Unsigned Divide	SBCD	Subtract Decimal with Extend
EOR	Exclusive OR	Scc	Set Conditional
EXG	Exchange Register	STOP	Stop
EXT	Sign Extend	SUB	Subtract
JMP	Jump	SWAP	Swap
JSR	Jump to Subroutine	TAS	Test and Set Operand
LEA	Load Effective Address	TRAP	Trap
LINK	Link and Allocate	TRAPV	Trap on Overflow
LSL	Loical Shift Left	TST	Test an Operand
LSR	Logical Shift Right	UNLK	Unlink

Table 2.6 Variations of instruction types

Instruction Type	Variation	Description	Instruction Type	Variation	Description
ADD	ADD	Add			to Status Register
	ADDA	Add Address	MOVE	MOVE	Move Source to Destination
	ADDQ	Add Quick		MOVEA	Move Address
	ADDI	Add Immediate		MOVEM	Move Multiple Registers
	ADDX	Add with Extend		MOVEP	Move Peripheral Data
AND	AND	Logical And		MOVEQ	Move Quick
	ANDI	And Immediate		MOVE from SR	Move from Status Register
	ANDI to CCR	And Immediate to Condition Codes		MOVE to SR	Move to Status Register
	ANDI to SR	And Immediate to Status Register		MOVE to CCR	Move to Condition Codes
				MOVE USP	Move User Stack Pointer
			NEG	NEG	Negate
				NEGX	Negate with Extend
CMP	CMP	Compare	OR	OR	Logical Or
	CMPA	Compare Address		ORI	Or Immediate
	CMPM	Compare Memory		ORI to CCR	Or Immediate to Condition Codes
	CMPI	Compare Immediate			
EOR	EOR	Exclusive Or		ORI to SR	Or Immediate to Status Register
	EORI	Exclusive Or Immediate			
	EORI to CCR	Exclusive or Immediate to Condition Codes			
	EORI to SR	Exclusive Or Immediate			
SUB	SUB	Subtract			
	SUBA	Subtract Address			
	SUBI	Subtract Immediate			
	SUBQ	Subtract Quick			
	SUBX	Subtract with Extend			

Figure 6 Instruction Set

from the disk drive. This then continues for the next floppy disk sector. The operating system modules that do this are the RBF file manager and the floppy disk driver.

Almost all the I/O functions of the MM/1 are under hardware interrupt control. These external interrupts are serviced in order of their priority, as shown in Table 4.

Interrupt Line	Priority	I/O Function
IN5	1	Floppy Drive Keyboard Serial Port /t1
IN4	2	Serial Port /t2 Serial Port /t3 Serial Port /t4 SCSI Bus
IN2	3	Parallel Ports
INT2N	3	VSC
UART Receive	5	Serial Port /t0
UART Transmit	6	Serial Port /t0
Inter-IC Bus	7	Inter-IC I/O

Table 4 Hardware Interrupt Priorities

The only I/O devices that are not serviced by a hardware interrupt are the sound I/O devices and the real time clock.

5.2.3 DMA

Direct Memory Access (DMA) is used on the floppy disks and SCSI bus controllers, as well as sound I/O. This allows sound files to be played or recorded without slowing down other CPU functions. It also means lightning fast disk transfers. DMA is controlled by the 68070 and, as shown in Table 5 there are two separate channels of differing priorities:

DMA Channel	Priority	Function
1	Highest	Sound I/O
2	Lowest	SCSI and Floppy I/O

Table 5 DMA Channels

The two DMA channels can operate independently of each other, but only one can have the bus at any one time. This allows software to pull sound files from the hard disk and play them without greatly affecting other functions like screen animation. Another advantage of DMA sound would be, for example, a game with a rather long music score that could be played over and over as the game progressed. Loading the entire sound file into memory would allow the DMA sound to be played continuously without affecting the game play at all.

Since Channel one has the higher priority, it can force Channel two to stop at the end of its current bus cycle and transfer control of the bus.

5.2.4 The Inter-IC Bus

The MM/1 is unique among personal computers in that it has built in communications hardware to control devices external to the unit via the high speed Inter-IC ("i-squared-c" or I2C) bus. The main principle of the bus is to connect a master device, such as a microcontroller or the 68070, and several slave systems.

There are a number of Inter-IC devices on the market and more and more are being designed into many consumer electronics devices such as VCR's, stereos, TV's and more. The table on the next page lists some of the devices already on the market.

The bus consists of two wires: a clock line and a serial data line, connecting up to 128 different Inter-IC devices. More than one device can be an active bus-master and more than one microprocessor can be connected to the bus and transfer data at the same time.
A typical Inter-IC bus is a ring configuration with the devices on the bus performing a variety of functions.

The List of Inter-IC Devices

Application-Dedicated ICs

Video/Radio

SAA3028	Infrared remote control transcoder (RC-5)
SAA1300	Tuner switching unit
SAA5240A	Computer-controlled Teletext (625-line British, German & Swedish)
SAA5240B	Computer-controlled Teletext (625-line British, German & Swedish)
SAA9020	Field memory controller
SAA9050/52	Digital NTSC/PAL color decoder
SAA9055	Digital SECAM decoder
SAA9061/62/63	Digital deflection decoder
SAA9068	Picture-in-picture controller
SAB3035/36/37	Digital tuning circuits for computer-controlled TV
SAA4700	Data line 16 decoder for the VCR
SAA1134/35	Same as SAA4700
TDA8400	Computer-controlled pre-scaler/synthesizer
TDA8405	Stereo decoder, West German TV system
TDA8432	Deflection processor and sync. controller
TDA8440	Peritelevision switch
TDA8443(A)	YUV/RGB matrix switch
TDA8461	PAL/NTSC color decoder and RGB processor
TEA6000/6100	FM/IF digital tuning IC for computer-controlled radio
TEA6057	PLL frequency synthesizer

Audio

TDA8420/21	Loudspeaker and headphone audio processor
TDA8425	Loudspeaker audio processor
TEA6300	Sound fader control and pre-amplifier/source selector
TEA6310T	Sound fader control with tone and volume control
PCF8200	Speech synthesizer (male/female speech)

Telecom

PCD3343	Microcontroller with 224 bytes RAM/3K ROM
PCD3348	Microcontroller with 256 bytes RAM/8K ROM
PCD3311/12	DTMF/tone/simplex-modem tone generator

Compact Disc

SAA1136	PCM-audio Ident-word Interface (IDI) for compact disc

Microcontrollers, Microprocessors

The CMOS PCF84CXX family

PCF84C00	256 bytes RAM/bond-out version for prototype development
PCF84C21	64 bytes RAM/2K ROM
PCF84C41	128 bytes RAM/4K ROM
PCF84C81	256 bytes RAM/8K ROM
PCF84C85	256 bytes RAM/4K ROM/extended I/O

The NMOS MAB84XX family

MAB8401	256 bytes RAM/bond-out version for prototype development
MAB8411	64 bytes RAM/1K ROM
MAB8421	64 bytes RAM/2K ROM
MAB8422	MAB8421 with reduced I/O: software PC
MAB8441	128 bytes RAM/4K ROM
MAB8442	MAB8441 with reduced I/O: software PC
MAB8461	128 bytes RAM/6K ROM

80C51-based microcontrollers

PCB83C552	256 bytes RAM/8K ROM ADC + additional functions
PCB83C652	256 bytes RAM/8K ROM

68000-based microprocessor

SCC68070	68000 CPU/MMU/UART/DMAs/timer

General-Purpose ICs

LCD drivers

PCF8566	96 Segment LCD driver/1:1 - 1:4 Mux
PCF8576	160 Segment LCD driver/1:1 - 1:4 Mux
PCF8577(A)	64 Segment LCD driver/1:1 - 1:2 Mux
PCF8578/79	Row/column LCD dot matrix driver/1:8 - 1:32 Mux

I/O Expanders

PCF8574(A)	8-bit remote I/O port (I²C-bus to parallel converter)
SAA1300	5-bit high current driver
SAA1064	4-digit LED driver
PCB8584	8-bit parallel to I²C-bus converter

Data Conversion

PCF8591	4-channel 8-bit MUX ADC/8-bit DAC
TDA8442	Quad 6-bit DAC
TDA8444	Octal 6-bit DAC

Memory

PCF8570	256 bytes static RAM
PCF8571	128 bytes static RAM
PCF8580/81/82	256 bytes EEPROM

Clock Timer

PCF8573	Clock/calendar
PCF8583	Clock/calendar/256 bytes RAM/.01 sec. resolution

Figure 7 Inter-IC Devices

Line	Meaning
SCL	Serial Clock Line
SDL	Serial Data Line

Table 6 Inter-IC Connections

Inter-IC Term	Definition
Transmitter	The device which sends data to the bus.
Receiver	The device which receives data from the bus.
Master	The device which starts a transfer, generates a clock signal, and ends a transfer.
Slave	The device addressed by the master.
Multi-Master	More than one master can attempt to control the bus simultaneously, but only one is allowed to do so. The message is not corrupted.
Synchronization	Procedure to force clock signals from two or more devices to occur simultaneously.

Table 7 Inter-IC Definitions

5.3 The Keyboard Interface

The keyboard used by the MM/1 is a "standard" IBM PC/XT type. Either an 88 or 101 key keyboard can be used as long as it is set for the XT mode. The reason for this is there is a difference in the data sent from an XT type and an AT type keyboard. Consider Figure 8 and Figure 9 below.

5.4 The Expansion Bus

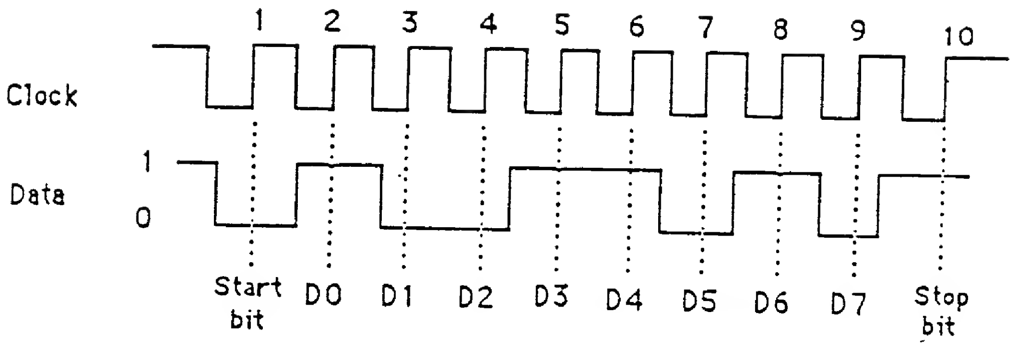


Figure 8 XT Type Scan Code

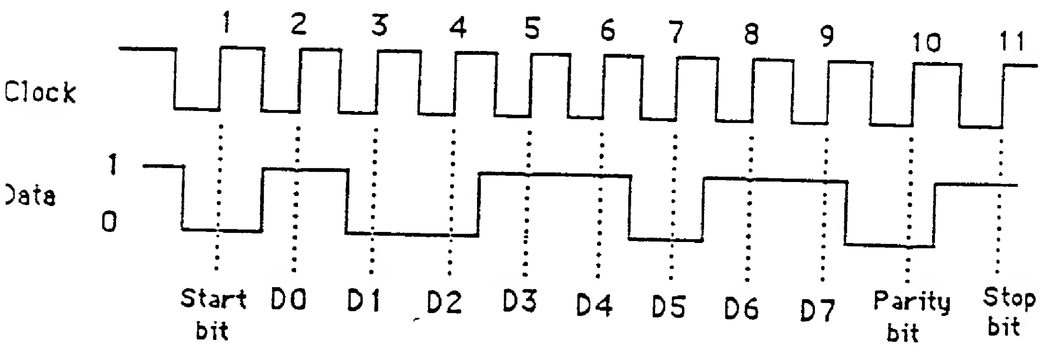


Figure 9 AT Type Scan Code

The MM/1 main processor board communicates with the additional I/O board through an expansion bus. This bus, consisting of a 32 position dual-header connector provides access to all the address and data lines from the CPU, as well as many other controlling lines that might be necessary for external I/O devices to use in signalling the CPU for attention. The table on the next page gives the pin-out for the bus.

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	GROUND	17	D14	33	REQ2*	49	A11
2	GROUND	18	D15	34	DONE*	50	A10
9	D0	19	UDS*	35	IRQ2*	51	A5
9	D8	20	LDS*	36	DRAM*	52	A8
5	D8	21	R/W*	37	A23	53	A8
6	D3	22	DTACK*	38	A22	54	A6
9	D8	23	CPUCLK	36	A21	55	A5
9	D5	24	RESET*	40	A2[illegible]	56	A4
9	D6	25	IRQ4*	41	A19	57	A3
10	D7	26	NMI*	42	A18	58	A2
11	D8	27	AS*	43	A17	56	A4
12	D9	28	BG*	44	A16	60	NMI*
13	D10	29	BR*	45	A15	61	+5V
14	D11	30	BGACK*	46	A14	62	+5V
15	D12	31	REQ1*	47	A13	63	+12V
16	D13	32	ACK1*	48	A12	64	+12V

* indicates an active low condition

Table 8 MM/1 Expansion Bus

End of Section

6 MM/1 SCSI System

The MM/1 SCSI drivers represent a new approach to disk system interface that provides more flexibility and functionality than before. MM/1 SCSI drivers support partitioned hard drives, variable sector sizes, multiple sector transfers, and write through caching. The section below is provided by Carl Kreider.

6.1 Bit-Map Sizes

The bit map size on any disk is limited to 64K bytes. There is a limit of 512K sectors that can be addressed by the bit map, or 128M bytes with 256 byte sectors, 256M bytes with 512 byte sectors, or 512M bytes with 1024 byte sectors. One of the old solutions to this problem was to change the cluster size from 1 to 2 or 4. In general, it is a better idea to run a larger sector size instead. The first benefit is faster disk I/O. The second benefit is less trouble with the dreaded error 217, or file segment list full. Using 512 byte sectors will more than double the number of segments. The third benefit is that you can put more data on the disk because less space is used for format overhead. The fourth benefit is that you can utilize economical drives that won't run anything but 512 byte sectors. The down side of sector sizes not equal to 256 bytes is that a lot of software assumes the old size of 256 bytes.

6.2 Partitions

Even with 512 byte sectors, the disk size limit with one sector clusters is 256M bytes. Partitioning is a nicer solution to this problem than running a larger cluster size for many of the same reasons as above. The mechanism is to increment the port address by one for each partition. For the MM/1, the port address is _$00E00000. This can be saved for a raw or whole device. Partition one is then _$00E00001, two is _$00E00002, and so on. There can be sixteen partitions all together, since the least four bits of the address are used for partitions. The PD_TotCyls field is set to the total cylinders on the drive while PD_CYL is set to the number of cylinders on the partition. PD_LSNOffs is set to the base LSN of the partition. For example, assume a 200MB drive with 409600 sectors 512 bytes each, arranged as 800 cylinders of 8 heads each and 64 sectors per track.

The parameters for multiple partitions are shown in Figure 10.

parm	partition 1	partition 2
address	_$00E00000	_$00E00001
size (MB)	100	100
PD_CYL	400	400
PD_LSNOffs	0	204800
PD_TotCyl	800	800

PARAMETERS FOR TWO PARTITIONS OF 100 MB EACH

parm	partition 1	partition 2	partition 3
address	_$00E00000	_$00E00001	_$00E00002
size (MB)	50	75	75
PD_CYL	200	300	300
PD_LSNOffs	0	102400	256000
PD_TotCyl	800	800	800

PARAMETERS FOR THREE PARTITIONS OF 50MB, 75MB, AND 75MB

Figure 10 Parameters for Multiple Disk Partitions

One needs to physically format the drive first with a descriptor that describes the whole disk. Then logically format each partition (but not the whole device descriptor if used). This will prepare the drive for use.

There are some new fields in the descriptor (or new flags in existing fields, depending on your perspective). These are PD_Cntl and PD_ScsiOpt. The only option currently allowed for PD_ScsiOpt is bit 3, the parity option. This setting must agree with the drive controller strapping. Of more interest is PD_Cntl. Bit 0 prevents formatting when set to one. This helps prevent accidental overwrite of the hard disk.

Keep a special descriptor around that must be loaded from disk with bit 0 set to zero for the rare times you format. Bit 1 enables multi-sector I/O when set to one. This will improve throughput. Bit 2 is set for any device with removable media, like Syquest 555 drives. Bit 3 is set for embedded SCSI drives, where the parameters of the drive are known to the controller. In this case, the fields in the descriptor that specify the drive geometry (PD_CYL, PD_SID, PD_SCT, PD_T0S, PD_ILV, PD_TotCyls) can be set to zero, since the drive will be queried by the driver for their values. Bit 4 is set if the device can format a single track. This is generally not embedded SCSI drives, but certain controllers can do this.

6.3 SCSI System Concept

The basic premise of this system is to break up the OS-9 driver into separate High-Level and Low-Level areas of functionality, so that different file managers and drivers can talk to their respective devices on the SCSI bus.

The "High-Level" functionality is handled by the device driver. This is the module that is called directly by the appropriate file manager. Drivers deal with all controller-specific/device-class issues. They prepare the command packets for the SCSI target device and then pass this packet to the "Low-Level" subroutine module.

This "Low-Level" module will pass the command packet (and data if necessary) to the target device on the SCSI bus. The low-level code does NOT concern itself with the contents of the commands/data, it simply performs requests on behalf of the high-level driver. The low-level module is also responsible for coordinating all communication requests between the various high-level drivers and itself.

The device descriptor module contains the name strings for linking the modules together. The file manager and device driver names are specified in the normal way. The "low-level" module name associated with the device is indicated via the "DevCon" offset in the Device Descriptor. This offset pointer will point to a string that contains the name of the low-level module.

6.4 Sample SCSI System

The example system setup below shows how drivers for disk and tape devices can be "mixed" on the SCSI bus, without interference:

6.4.1 Hardware Configuration - 2 SCSI Devices

Teac MT2-ST Tape Drive with Embedded SCSI Controller:

- addressed as SCSI ID 4

Seagate 296N Hard Disk with Embedded SCSI Controller:
- addressed as SCSI ID 0

Host CPU: MM/1
- uses WD33C93 SBIC Interface chip.
- "own id" of chip is SCSI ID 7

The hardware setup would look like this:

```
SCSI Bus:          ------------------------------>
                               |              |
                               |              |
SCSI Controllers:           MT2-ST         ST296N
                               |              |
                               |              |
Physical Devices:            Tape            H/D
```

6.4.2 Software Configuration - 2 SCSI Devices

The "high-level" drivers associated with this configuration are:
- SBTEAC: handles Teac tape device
- RBVCCS: handles Seagate ST296N SCSI hard disk

The "low-level" module associated with this configuration is:

- SCSI_MM1: handles WD33C93 Interface on the MM/1 CPU

A conceptual map of the OS-9 Modules for this system would thus look like this:

```
Kernel Level:                   OS-9 Kernel
                                     |
                          ----------------------
                          /                    \
File Manager Level:    RBF (disks)          SBF (tapes)
                          |                    |
Device Driver Level:   RBVCCS                SBTEAC
                          \                    /
                          ----------------------
                                     |
                                 SCSI_MM1
```

This permits expansion of the hardware system to be performed easily. For example, let us add an Adaptec ACB-4000 Disk Controller to the SCSI bus on the MM/1.

6.4.3 Hardware Configuration - 3 SCSI Devices

The added hardware configuration is:

Adaptec ACB-4000 Controller:
- addressed as SCSI ID 1
- hard disk addressed as controller's LUN 0

The hardware setup would thus look like this:

```
               SCSI Bus:     ---------------------------------------------->
                                |              |              |
                                |              |              |
     SCSI Controllers:       MT2-ST         ACB-4000       ST296N
                                |              |              |
                                |              |              |
     Physical Devices:       Tape           H/D            H/D
```

6.4.4 Software Configuration - 3 SCSI Devices

The added "high-level" driver associated with this configuration is:

- RB4000: handles hard drive on the ACB-4000.

The conceptual map of the OS-9 Modules for the system would now look like this:

```
Kernel Level:                     OS-9 Kernel
                                       |
                             ----------------------
                            /                      \
File Manager Level:     RBF (disks)          SBF (tapes)
                          /   \                    |
                         /     \                   |
Device Driver Level: RBVCCS   RB400             SBTEAC
                          \     \                 /
                             ----------------------
                                       |
                                   SCSI_MM1
```

Other drivers and devices such as CD-ROMs, laser printers, or even Ethernet cards could be added to the I/O system.

6.5 Variable Sector Programming Note

Because of the variable sector size capability used in OS-9/68000 Version 2.4 and above, programs that control the read/write pointer by directly seeking to disk locations will not work if the calculations are based upon a 256 byte sector size. To correct this problem, all such programs much be reworked to check the PD_SSize field of the device descriptor to get the medias logical sector size and use that figure for sector calculations. Consult your OS-9/68000 technical documentation for more information.

6.6 SCSI Physical Characteristics

SCSI devices are daisy-chained together using a common cable. Both ends of the cable are terminated. All signals are common between all SCSI devices.

6.7 SCSI Bus Communication

Communication on the SCSI bus is allowed between only two SCSI devices at any given time. There is a maximum of eight SCSI devices. Each SCSI device has a SCSI ID bit assigned as shown in Table 10 below:

DB(7)	DB(6)	DB(5)	DB(4)	DB(3)	DB(2)	DB(1)	DB(0)
|	|	|	|	|	|	|	ID=0
|	|	|	|	|	|	ID=1	
|	|	|	|	|	ID=2		
|	|	|	|	ID=3			
|	|	|	ID=4				
|	|	ID=5					
|	ID=6						
ID=7							

Table 10 SCSI ID Bits

Pin Number	Signal	Active Low	Pin Number	Signal	Active Low
2	DB(6)	YES	28	GROUND	
4	DB(1)	YES	32	GROUND	
6	DB(2)	YES	32	ATN	YES
8	DB(3)	YES	34	GROUND	
14	DB(4)	YES	36	BSY	YES
12	DB(5)	YES	32	ACK	YES
14	DB(6)	YES	40	RST	YES
16	DB(7)	YES	42	MSG	YES
18	DB(P)	YES	44	SEL	YES
20	GROUND		46	C/D	YES
22	GROUND		48	REQ	YES
24	GROUND		50	I/O	YES
26	TERMPWR				

Table 9 SCSI Pin Assignments

When two SCSI devices communicate on the SCSI bus, one acts as an initiator and the other acts as a target. The initiator originates an operation and the target performs the operation. A SCSI device usually has a fixed role as an initiator or target, but some devices may be able to assume either role.

An initiator may address up to eight peripheral devices that are connected to a target. An option allows the addressing of up to 2,048 peripheral devices per target using extended messages.

Certain SCSI bus functions are assigned to the initiator and certain SCSI bus functions are assigned to the target. The initiator may arbitrate for the SCSI bus and select a particular target. The target may request the transfer of COMMAND,

DATA, STATUS, or other information on the DATA BUS, and in some cases it may arbitrate for the SCSI bus and reselect an initiator for the purpose of continuing an operation.

Information transfers on the DATA BUS are asynchronous and follow a defined REQ/ACK handshake protocol. One byte of information may be transferred with each handshake. An option is defined for synchronous data transfer.

6.8 SCSI Bus Signals

There are a total of eighteen signals. Nine are used for control and nine are used for data. (Data signals include the parity signal option). These signals are described in Table 11.

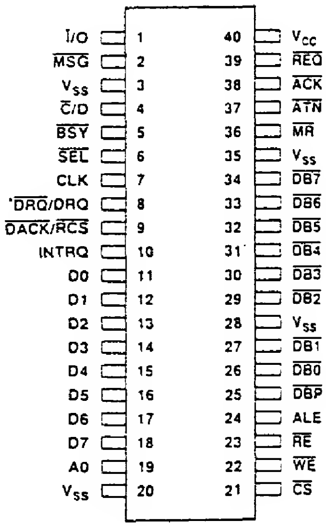


Figure 11 WD33C93A SCSI Controller

BSY (BUSY).	An "OR-tied" signal that indicates that the bus is being used.
SEL (SELECT).	A signal used by an initiator to select a target or by a target to reselect an initiator.
C/D (CONTROL/DATA).	A signal driven by a target that indicates whether CONTROL or DATA information is on the DATA BUS. True indicates CONTROL.
I/O (INPUT/OUTPUT).	A signal driven by a target that controls the direction of data movement on the DATA BUS with respect to an initiator. True indicates input to the initiator. This signal is also used to distinguish between SELECTION and RESELECTION phases.
MSG (MESSAGE).	A signal driven by a target during the MESSAGE phase.
REQ (REQUEST).	A signal driven by a target to indicate a request for a REQ/ACK data transfer handshake.
ACK (ACKNOWLEDGE).	A signal driven by an initiator to indicate an acknowledgment for a REQ/ACK data transfer handshake.
ATN (ATTENTION).	A signal driven by an initiator to indicate the ATTENTION condition.
RST (RESET).	An "OR-tied" signal that indicates the RESET condition.
DB(7-0,P) (DATA BUS).	Eight data-bit signals, plus a parity-bit signal that form a DATA BUS. DB(7) is the most significant bit and has the highest priority during the ARBITRATION phase. Bit number, significance, and priority decrease downward to DB(0). A data bit is defined as one when the signal value is true and is defined as zero when the signal value is false.
Data parity DB(P).	Parity is odd. The use of parity is a system option (i.e., a system is configured so that all SCSI devices on a bus generate parity and have parity detection enabled, or all SCSI devices have parity detection disabled or not implemented).

Table 11 SCSI Bus Commands

End of Section

7 Hard Disk Drives

The MM/1 uses hard disk drives with the SCSI (Small Computer Systems Interface) standard. The SCSI standard is rapidly becoming widely used in the industry and SCSI hard drives are becoming very cost competitive with other non-SCSI drives. The SCSI system is actually a high-speed parallel interface arraigned in a "bus" topography. Up to seven SCSI devices can be installed on this bus. Any type of hard disk drive with an embedded SCSI controller will work in any of the common sizes with no known upper limit. The minimum size suggested is 30 megabytes.

7.1 Installing the Drive

To install a hard disk drive in the MM/1 case you need to refer to the drive manufacturers documentation. Make sure the drive is setup as a SCSI device zero (0) and no parity checking is selected. Usually, these options are set on the drive using jumpers. Make sure you mount the drive in a position where the SCSI connector cable can reach the MM/1 circuit boards.

Plug the 50 pin SCSI cable you should have received with the drive into the SCSI connector on the I/O board. The cable is keyed so it will only go in one way. DO NOT force it into the connector. Plug the other end of the cable into the SCSI hard drive which is also keyed. Plug in the disk drive power cable and you're ready to proceed.

7.2 Formatting Hard Drives

If you have installed a hard disk drive, you must format it before it can be used. However, formatting hard drives is different with the MM/1 than what many users might be used to from previous experience with a CoCo hard drive system or an IBM clone system.

The biggest difference between a SCSI hard drive and the more widely known ST406 interface hard drives is that the drive itself is "intelligent". What this means to you the user, is that you do not need to know all the many parameters of the SCSI hard disk drive to format it. All you need to know is the total number of sectors on the drive.

For example, the drive types and their respective number of sectors in Table 12 was taken from a Seagate handbook.

Drive Type	Number of Sectors	Capacity
ST138N	63,139	32 MB
ST157N	95,015	48 MB
ST225N	41,720	21 MB
ST251N	84,254	43 MB
ST277N	126,790	64 MB

Table 12 Seagate Drive Parameters

So, to correctly format a SCSI drive, all you need to tell the format utility is the number of sectors on the drive. However, the OS-9/68000 format utility uses cylinders and heads as drive size parameters. So which is right?

First, you call the format utility thus:

```
_$ format -nvnp -v=Test /h0
```

The format utility will do a logical format of the hard drive creating the sector allocation table and right the information in logical sector zero (LSN0). It will return, among other things, the total number of sectors created on the drive. This is the number we are interested in.

If the number of sectors returned is different than the total number of sectors on your hard disk, then all you need to do is use the DMODE utility to change ANY parameter about the drive size (such as number of cylinders, number of heads, number of sectors per track) to increase or decrease the number of sectors that will be formatted on that drive until it equals the number of actual sectors.

7.2.1 Using DMODE

For example, you have a ST225N drive. You run the format utility as shown above and it returns 41,680 sectors. You would not need to change anything since the parameters are set correctly. However, if you have an ST251N drive, you would need to DOUBLE the number of sectors that the format utility will create. Therefore, you could take the number of cylinders OR the number of heads and double that value. Using the DMODE utility, you might try:

```
_$ dmode /h0

drv=00 stp=00 typ=80 dns=00 cyl=026C sid=04 vfy=01 sct=0011
tos=0011 ilv=03 sas=11 tfm=00 toffs=00 soffs=00 ssize=0200
cntl=0003  trys=07  lun=00  wpc=0000  rwr=0000  park=0000
lsnoffs=00000000
```

We want to double the number of heads (sides) to double the number of sectors:

```
_$ dmode /h0 sid=8

drv=00 stp=00 typ=80 dns=00 cyl=026C sid=08 vfy=01 sct=0011
tos=0011 ilv=03 sas=11 tfm=00 toffs=00 soffs=00 ssize=0200
cntl=0003  trys=07  lun=00  wpc=0000  rwr=0000  park=0000
lsnoffs=00000000
```

Alternatively, we can double the number of cylinders:

```
_$ dmode /h0 cyl=04d8

drv=00 stp=00 typ=80 dns=00 cyl=04D8 sid=04 vfy=01 sct=0011
tos=0011 ilv=03 sas=11 tfm=00 toffs=00 soffs=00 ssize=0200
cntl=0003  trys=07  lun=00  wpc=0000  rwr=0000  park=0000
lsnoffs=00000000
```

Once we have the number of cylinders that format will create matching the total number of sectors actually on the hard drive, you can then format the drive without any command line options. This will then perform a physical format, ask you for a volume name, and then optionally perform a verify (which you will want to do with a new drive).

Finally, you DO NOT need to save the changed hard drive descriptor to be able to access that drive in the future. As was mentioned before, SCSI hard drives are "intelligent" drives.

7.3 Using Two Hard Drives

The SCSI bus on the MM/1 is designed to easily add additional devices to expand your system. Adding a second hard drive is therefore very easy. The first hard drive on the system should have been selected as SCSI device zero (0). Your second hard drive should then be selected as SCSI device one (1). Consult the drive documentation to see how this is done.

MAKE SURE TO REMOVE ANY TERMINATING RESISTORS ON EITHER THE FIRST OR THE SECOND HARD DRIVE. This is imperative if you do not wish to severely damage or destroy your MM/1. Only one drive can have terminating resistors on it--it doesn't matter which one.

After mounting the second drive in your case, you'll need to get a new cable that includes an additional 50 pin header connector to plug into the second drive.

Once the drive is mounted, boot the system. If everything was setup correctly, it will boot as before. If the system does not boot, you probably didn't set the second drive to SCSI device one, and it thinks it is device zero, so the system is trying to boot from an empty drive. This all depends upon where the second drive appears to the bus.

If the system boots normally, you must then create a device descriptor for the second drive. The easiest way to do this is to save the existing descriptor for /h0 and rename it /h1. Now, edit the bootlist file including the new /h1 descriptor and OS9Gen a new bootdisk. If you IDENT the new bootfile, you'll see there are two /h0 descriptors in the file. This is fine. Now, you can use the MODED utility on the new bootfile like so:

```
        _$ chd /d0 (or whatever is you boot device)
        _$ moded

OS-9/68000 Module Editor
Copyright 1987 Microware Systems Corp.
Type ? for editing help message

moded: f   <-- type "f"
filename to use: os9boot  <-- type "os9boot"

moded: e   <-- type "e"
name of module to use: h0
```

At this point, MODED will begin displaying the fields within the first /h0 descriptor it finds in the bootfile. The first item to change is the descriptor name to "h1":

```
moded shows: descriptor name       :  h0 =
you type : h1

press the ENTER key until

moded shows: drive number          :   0 =
you type 1

press the ENTER key until

moded shows: scsi controller id   : _$00 =
you type _$01

press the ENTER key once more and moded displays:

moded: q   <-- you enter "q"
```

MODED now prompts you to press 'y' to write out the changes and you do.

At this point, enter "break" to reboot your system. When it comes up again, use MDIR to verify that a h1 module is present. If so, then you can begin formatting the second drive as you did the first.

7.3.1 Adding More Hard Drives

If the need arises to add even more hard drives to the SCSI bus, all you need to remember is to increment the SCSI device numbers as you add a new device, and to make sure no SCSI parity is set on the drive. Of course, a 50 conductor cable with suitable connectors will have to be bought or made. External drives can be added to the system by using the back panel SCSI connector (available from IMS) and using a separate drive and power supply. Once the drive is physically installed, process as you did when adding the second SCSI drive.

7.4 Some Considerations

Some SCSI hard drives tested with the MM/1 had some trouble being accessed when the system first boots after a power up. This is not necessarily a problem with the MM/1 or its design, but rather a quirk in the SCSI implementation on some drives. This problem will appear, if at all, when you first turn on the system and it goes to run startup (or boot) from the hard drive. It will just hang. Resetting the system or just trying to access the drive a second time will cause everything to work normally.

Some SCSI drives take longer to initialize themselves. In some cases, the drive will not be ready to receive commands when the MM/1 tries to access it when the system is booting. In this case, the system will also hang, but, you should be able to chd to the drive and work from there.

If this happens, you can change the delay time the MM/1 will use before trying to access the default device during bootup. You do this by using the MODED utility on the INIT module in your OS9Boot file. As described above, run MODED and select the file to read as the OS9Boot file on your default startup device (normally /d0). Then select the INIT module as the module to edit. Step through the fields until the field titled "coldstart chd retry count" is reached. Change the number for this field to 50 or more, then exit MODED and reboot the system. If it still does not chd to the default device and run, try making the delay count larger.

End of Section

8 Floppy Disk Drives

The MM/1 comes with one built-in 1.44 Megabyte 3.5 inch floppy disk drive, and the capacity for adding 3 more of differing types, although only 5.25 and 3.5 inch drives are supported.

8.1 Physical Interface

The floppy disk drive interface is an SA400, or "IBM PC" type interface. See Table 13 below. A single 34 conductor ribbon cable provides all the required signals to the drive with the exception of power which is supplied by a standard "IBM PC" type drive power connector.

The 34 conductor cable is pinned straight through to each drive. Unlike the IBM-PC system, no "twisting" of the cable is needed. Make sure each drive on the chain is properly selected, i.e., drive /d0 is selected as drive 0, drive /d1 is selected as drive 1, and so on.

8.1.1 Pin 2 Jumper Settings

One jumper setting on the main processor controls the density pin of the disk interface (pin 2). This pin is used in several different ways by various drive manufacturers. Basically, the pin can either:

1. Determine the read/write head current going to drive

   or

2. Determine the drive's RPM for dual speed (300/360) drives

To provide the user with the ability to interface a large number of available disk drives, jumper P13 on the processor provides either a logical high or low signal to pin 2 of the floppy interface. Setting jumper pins 1 and 2 provide the logic high setting, while setting jumper pins 2 and 3 will provide a logic low. Depending upon how the individual drive manufacturer uses this pin, the user may or may not need to jumper it.

8.1.2 Terminating Resistors

Pin Number	Signal	Active Low Signal
2	See Text	
4	Not Connected	
[illegible]	Drive Sel 3	Yes
8	Index	Yes
10	Drive Sel 0	Yes
12	Drive Sel [illegible]	Yes
14	Drive Sel 2	Yes
16	Motor On	Yes
16	Direction	Yes
20	Step	Yes
22	Write Data	Yes
24	Write Enable	Yes
26	Track 0	Yes
28	Write Prot	Yes
30	Read Data	Yes
32	Head Select	Yes
34	Drive Ready	Yes

Table 13 SA400 Interface

Most 5.25 inch floppy disk drives require a 150 ohm pull-up resistor in the drive chain to insure correct logic levels during operation. Since the floppy disk controller used in the MM/1 (WD37C65/A) is TTL compatible, this is not always necessary. Many 3.5 inch drives, however, use CMOS technology for low power consumption

in portable computers. If using a CMOS 3.5 inch drive, pull-ups of 150 ohms or greater are required on the drive side. Most 3.5 drives have internal 1 Kohm pull-ups so this is an easy requirement to fulfill. The reason for this is to insure the Western Digital specification for sink current is not exceeded. Failure to provide pull-ups might severely damage the chip internals.

8.2 Disk Drive Settings

The OS-9/68000 operating system is unique in the method used to define and install new devices. The ease in which this can be done is unparalleled in any other operating system. By simply changing a few parameters in a disk device descriptor module, even while it is in memory, you can setup your disk drives to read many differently formatted disks.

8.2.1 DMODE Utility

If you issue the DMODE command on one of your floppy disk drive modules, you might see something like this:

```
name=d0
drv=0 stp=3 typ=$21 dns=$03 cyl=80 sid=2 vfy=0 (on) sct=33
t0s=33sas=8   ilv=2   tfm=0   toffs=0   soffs=0   ssize=256
cntl=_$0000  trys=7  lun=0  wpc=0  rwr=0  park=0  lsnoffs=0
totcyls=80 ctrlrid=0 rates=_$00 scsiopt=_$0000
```

To change any of the parameters shown above, one would use either the MODED utility, or the easier DMODE utility supplied with your MM/1. You use this utility in the same manner as the XMODE or TMODE utilities explained in the Microware manuals. For example, to change the number of cylinders on a drive, you would do the following:

```
_$ dmode /d0 cyl=40
```

or

```
_$ dmode /d0 cyl=80
```

Those parameters that require input in hexadecimal numbers are preceded with a dollar sign (_$).

8.3 Disk Drive Parameters

Out of the many parameters shown in the DMODE utility output, only a few are of interest for this section. These are explained below:

Typ	Device type - 5" or 3.5", hard or floppy
sct	Number of sectors per track
t0s	Number of sectors per track on track 0
toffs	Track base offset
soffs	Sector base offset
ssize	Sector size in bytes
rates	Transfer rates and rotational speed

Table 14 shows some of the more common disk drive formats used by various OS-9 machines and the parameters that you need to change to read/write these disks (courtesy of Ed Gresick at DELMAR Co.):

Drive	Size	Trks	Typ	sct	t0s	toffs	soffs	ssize	rates
MM/1	3.5	80	_$21	33	33	00	00	256	_$10
Atari ST	3.5	80	_$26	16	16	00	00	256	_$10
CoCo	3.5	80	_$26	16	16	00	01	256	_$10
CoCo	5.25	80	_$24	18	16	00	01	256	_$21
CoCo	5.25	40	_$20	18	16	00	01	256	_$21
Mizar	3.5	80	_$26	16	16	00	00	256	_$10
Mizar	5.25	80	_$24	16	16	00	00	256	_$21
OSK Standard	3.5	80	_$06	16	10	00	00	256	_$10
OSK Standard	5.25	80	_$04	16	10	00	00	256	_$21
OSK Universal	3.5	80	_$26	16	16	01	00	256	_$10
OSK Universal	5.25	80	_$24	16	16	01	01	256	_$21

CYL AND TOTCYL SHOULD BE SET EQUAL TO TRKS

Table 14 Disk Drive Formats

End of Section

9 Serial Ports

The MM/1 basic system comes with one ready to go serial port, and one that requires an external paddle board to act as a second serial port. The extended system board adds to that one more ready to go port, and connections for two more ports using the external paddle boards. The external paddle boards are nothing more than serial line drivers that take the 5 volt TTL logic signals and boost them to RS-232 levels via a Motorola MC145407 chip.

The following description of the serial ports is provided by Carl Kreider, OSK driver writer of global renown.

9.1 Physical Description

All of the five serial ports are not the same. The chart below should give you an idea of the differences:

It would seem that /t3 and /t4 are similar and in fact that is the case. Both reside in the same chip - a dual UART (DUART). It would also seem that /t1 and /t2 are similar, but there are differences. T1 can only support a three wire serial connection. There are no modem control lines.

The differences in the number of modem control lines supported by each port is due to the architecture of the MM/1. Ports /t3 and /t4 are designed to work with modem high speed modems that require hardware handshaking to work correctly. Ports /t0 and /t2 make excellent ports for serial mice or external terminals.

9.1.1 Signalling on DCD Transition

The ports all (except for t1) support signals on DCD transition. You select which transition (on to off or off to on) you wish to be signalled on. The signal clears when sent, so it needs to be reset after reception. The C bindings are in the C manual under _ss_dcoff() and _ss_dcon(). The assembler bindings are listed in the OS-9 System Calls section of the Operating System manual under I_$SetStt SS_DCOff/SS_DCOn. Note that for /t0 and /t2 you must be in the software flow control mode (i.e., XON/XOFF enabled in the descriptor -- see Hardware Handshaking below).

Port	Controller	Modem Lines	Max Baud
/t0	68070 internal port	DTR, CD	19,200
/t1	68901 on cpu board	None	19,200
/t2	68901 on I/O board	DTR, CD, RTS	19,200
/t3	68681 on cpu board	DTR, CD, CTS, RTS	34,800
/t4	68681 on I/O board	DTR, CD, CTS, RTS	34,800

Table 15 Serial Port Characteristics

9.1.2 RTS Control

The ports all (again, except for t1) support software control of the RTS line. The C bindings are in the C manual under _ss_dsrts() and _ss_enrts(). The assembler bindings are listed in the OS-9 System Calls section of the Operating System manual under I_$SetStt SS_DsRTS/SS_EnRTS. Note that for /t0 and /t2 you must be in the software flow control mode.

9.1.3 Hardware Flow Control

The ports all (except for t1) support both hardware and software flow control. There is a distinction between hardware and software flow control modes. Hardware handshaking is selected by setting the high bit on the type byte in the descriptor. (Note that no software exists to do this dynamically. The change can only be made by patching the boot file or reassembly.) In the hardware flow control mode, the state of the CTS line is sampled before transmission of the next character. If it is false, the character will not be sent. This is done with hardware on /t0, /t3 and /t4 but is done is software on /t2.

Please note that on /t0 and /t2 one connection serves the purpose of both CTS and CD making CD signalling and hardware flow control mutually exclusive. Ports /t3 and /t4 do not share function, but the CTS line will be ignored unless the hardware mode is selected so that it functions in a similar manner to the others. In the software flow control mode, CD can be used to signal CD change and xon/xoff will be used (if enabled in the descriptor) for flow control. In the hardware flow control mode, the state of the RTS line is controlled by how full the inbound buffer is. RTS is asserted when the port is opened or inized and remains on until there are less than 10 empty spots in the inbound buffer, when it will be negated. It will remain off until there are only 10 chars left to consume in the inbound buffer, when it will be asserted again. It will be negated when the port is closed. Please note that on /t0 and /t2 one connection serves the purpose of both RTS and DTR. On /t3 and /t4, DTR is asserted at iniz or open and remains true until the port is closed.

In the software flow control mode, XON/XOFF (if enabled in the descriptor) will be used for flow control. DTR will be asserted at iniz or open and will remain true until the port is closed, when it will be negated. In addition, on ports /t0 and /t2 the line can be manually controlled with setstat for RTS. On /t3 and /t4, DTR will remain on, but RTS can be controlled in the same manner. Please note that t1 does support software flow control (if enabled in the descriptor).

9.1.4 Maximum Baud Rate Limitations

In some cases the serial ports may drop characters at maximum baud rates if disk or other activity keeps the interrupt masked for too long. This will not normally be a problem if the incoming data is being saved to a Ramdisk, but access to a hard drive, and especially a floppy disk drive will cause characters to be lost.

The solution to this is to either reduce the speed at the port, or just don't save any incoming data. Ports /t0 through /t2 will not exhibit this problem at speeds of 9600 baud or less. Ports /t3 and /t4 have a four character receive FIFO and are less susceptible to this problem. These ports can run unfettered at the maximum baud of 38,400 bps.

End of Section

10 I/O Devices

Details of the individual chips used in the MM/1 are not possible in this document. The MM/1 design uses a number of large scale integration devices that are far more complex than previous designs. To give a complete description of the operation of each device would mean reproducing the entire technical manual for each device in these pages.

Because of the complexity of the devices, the data required to make full use of them can only be had by having more than a partial description of their functions. Therefore, this manual will only explain the features that are unique to the MM/1 design and leave the serious hacker to obtain the necessary technical docs from the chip manufacturers. A list of manufacturers, with addresses and phone numbers is provided in an appendix to this document.

10.1 Base Addresses

The various I/O devices on the MM/1 are memory mapped for easy access by the different drivers. Table 16 shows the devices and their BASE addresses:

10.2 MC68901 Multi-Function Peripheral

In the MM/1, a single MC68901 is used on the processor board and one on the I/O board to provide serial ports as well as control functions for other devices on the board. For example, the following table shows the functions controlled by the processor board MC68901 by manipulating the Data Direction Register (DDR).

7	6	5	4	3	2	1	0
Keybd IRQ	Modem DTR	Floppy IRQ	Motor On	Master Slave	Carrier Detect	Drive Ready	Debug

Table 17 MC68901 DDR

Device	Address
MC68901 (processor board)	_$09FFC00
WD33C93 SCSI Controller	_$0E00000
AD7569 Sound DAC	_$0E00100
MC68230 PIA (Parallel Ports)	_$0E00180
DS1287 RTC	_$0E00200
MC68681 DUART	_$0E00280
MC68901 (I/O Board)	_$0E00380

Table 16 MM/1 I/O Device Addresses

By setting the appropriate bit in the DDR, each bit can be configured as an input or output. When a bit is written as a zero, the corresponding bit of the GPDR (General Purpose Data Register) remains in the high impedance state. Writing a one will cause that bit to be a push-pull output.

This condition is used to control the functions in table @na.@nd above. For example, keyboard IRQs can be turned off by masking bit 7, as can floppy disk IRQs by setting bit 5 to zero.

Accessing the modem control lines is through the GPDR. Setting and resetting bit 6 will toggle the DTR line at port /t0. Reading the same register and masking to detect bits 2 and 6 will give the status (set or not) for CD and DTR respectively. Masking with _$04 will yield TRUE if CD is set and FALSE if it is not. Of course, you can also toggle the status of CD in the same manner.

The second 68901 on the optional I/O board is addressed in the same manner as described above, except for the base address. Accessing the modem control lines on this port works the same as described above, except the control lines connect to different input pins. On this port, DTR is set or reset by toggling bit 1 of the GPDR and CD is set or reset by toggling bit 0.

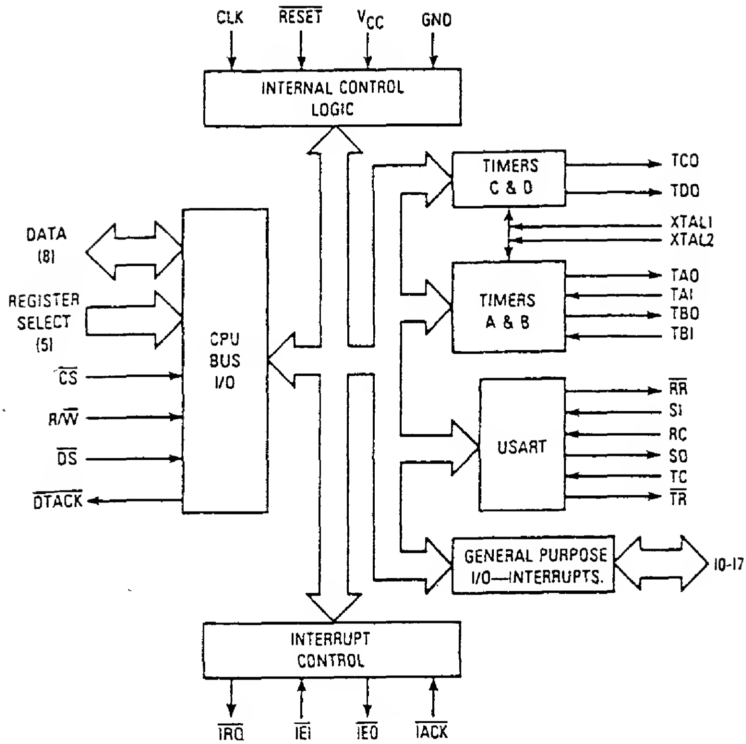


Figure 12 MC68901 Block Diagram

Masking the GPDR with a _$01 yields the status of CD.

On this port, you must be careful not to change the status of bits 7, 6, or 5 of the GPDR since these control the sound mode, SCSI reset and SCSI IRQ respectively. Bit 4 is not used so this could be hacked to control something in the future. Bits 2 and 3 are for the joystick fire buttons.

All registers of the MC68901 are accessed on odd byte boundaries. Accessing even boundary bytes causes a bus error. Save all registers that will be changed before writing to them so they can be restored when the operation is complete.

10.3 MC68681 DUART

One of these devices exists on the I/O board providing serial ports /t3 and /t4 as its only function. These serial ports differ from the other three in that /t3 and /t4 have a maximum baud rate of 34800 bits per second. This is the maximum rate supported in the serial driver for this device.

The MC68681 can be programmed to attain speeds of 1 Megabit per second in the two independent ports, has an 6-bit input and an 8-bit output port for control functions. These extended functions may be of interest to driver writers for extremely fast serial I/O applications.

No special features are used in the MM/1 for this device. It serves only as a dual serial port. The tables below explain the two parallel/control port functions:

7	6	5	4	3	2	1	0
N/C	N/C	N/C	N/C	Chan B DTR	Chan A DTR	Chan B RTS	Chan A RTS

8-Bit Output Port

7	6	5	4	3	2	1	0
1	1	N/C	N/C	Chan B DCD	Chan A DCD	Chan B CTS	Chan A CTS

6-Bit Input Port

Table 18 DUART Port Functions

Note that the input port is a 6-bit port. When this port is read, bits 6 and 7 will always return a 1. Bits marked as N/C are not connected within the MM/1, leaving them open for all sorts of possibilities.

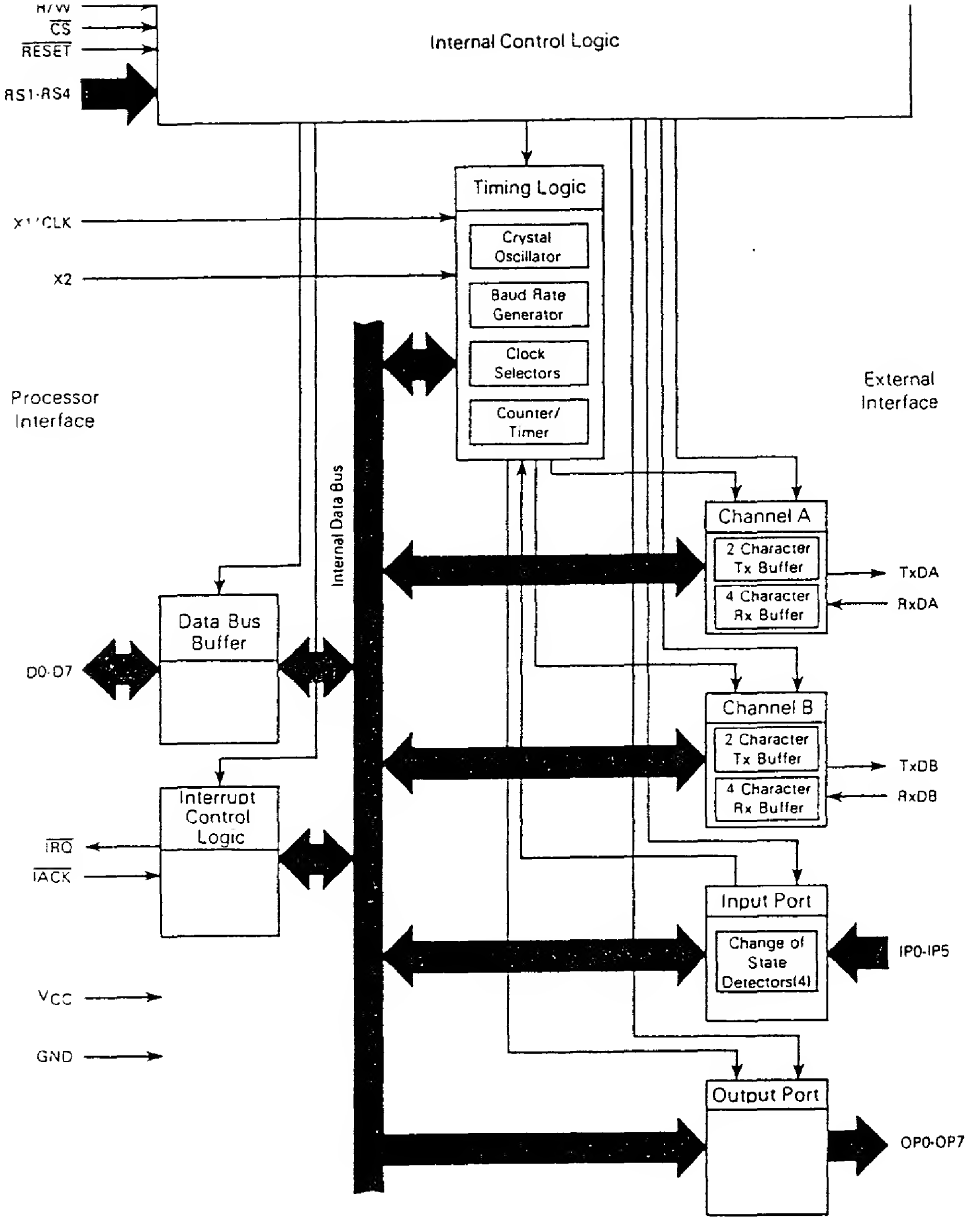


Figure 13 DUART Block Diagram

The device is capable of sending a hardware BREAK signal through the serial data lines. This feature is enabled through a _ss_brk() system call.

10.4 MC68230 Parallel Interface/Timer

The MC68230 on the MM/1 I/O board is more than just a parallel printer port interface. This device is a high-speed parallel I/O chip that can provide a multitude of functions given the proper driver. For example, consider the programmable modes in Table 19.

Mode 0	8-bit unidirectional on each port
Mode 1	16-bit unidirectional both ports
Mode 2	8-bit bidirectional on each port
Mode 3	16-bit bidirectional both ports

Table 19 68230 Operating Modes

As you can see, the 16-bit bidirectional mode can be used for some super quick I/O, process control, or whatever one could imagine. Since there are four programmable handshaking lines, two for each port, and DMA for each port can be enabled for input as well as output, the possibilities are almost endless.

The MC68230 general purpose control port consists of 8 control lines that function in much the same was as the general purpose data register on the MC68901. The only control function performed by this port is by using bit 0 as the sound or joystick I/O control.

10.5 Dallas DS1287 Real-Time Clock

This device is a direct replacement for the Motorola MC146818A RTC. It contains a lithium battery that is rated to last for at least 10 years, a time-of-day clock, a one hundred year calendar, and 50 bytes of non-volatile RAM.
Following is the table of offsets from the base address where the data from the RTC can be read.

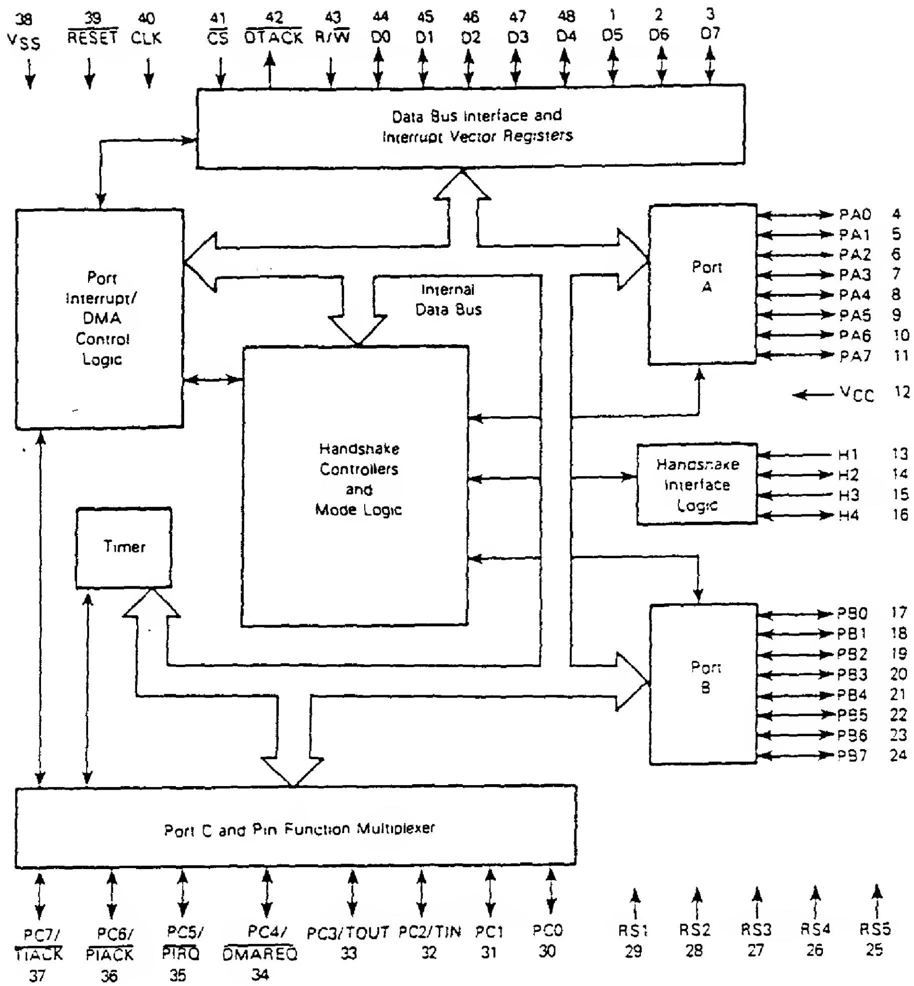


Figure 14 Parallel Interface Block Diagram

Beginning at offset _$0E is the 50 bytes of non-volatile RAM. This RAM can be used for just about anything since it is not presently used by the MM/1 or OS-9/68000. If an application for this RAM does come up, it will of course cause

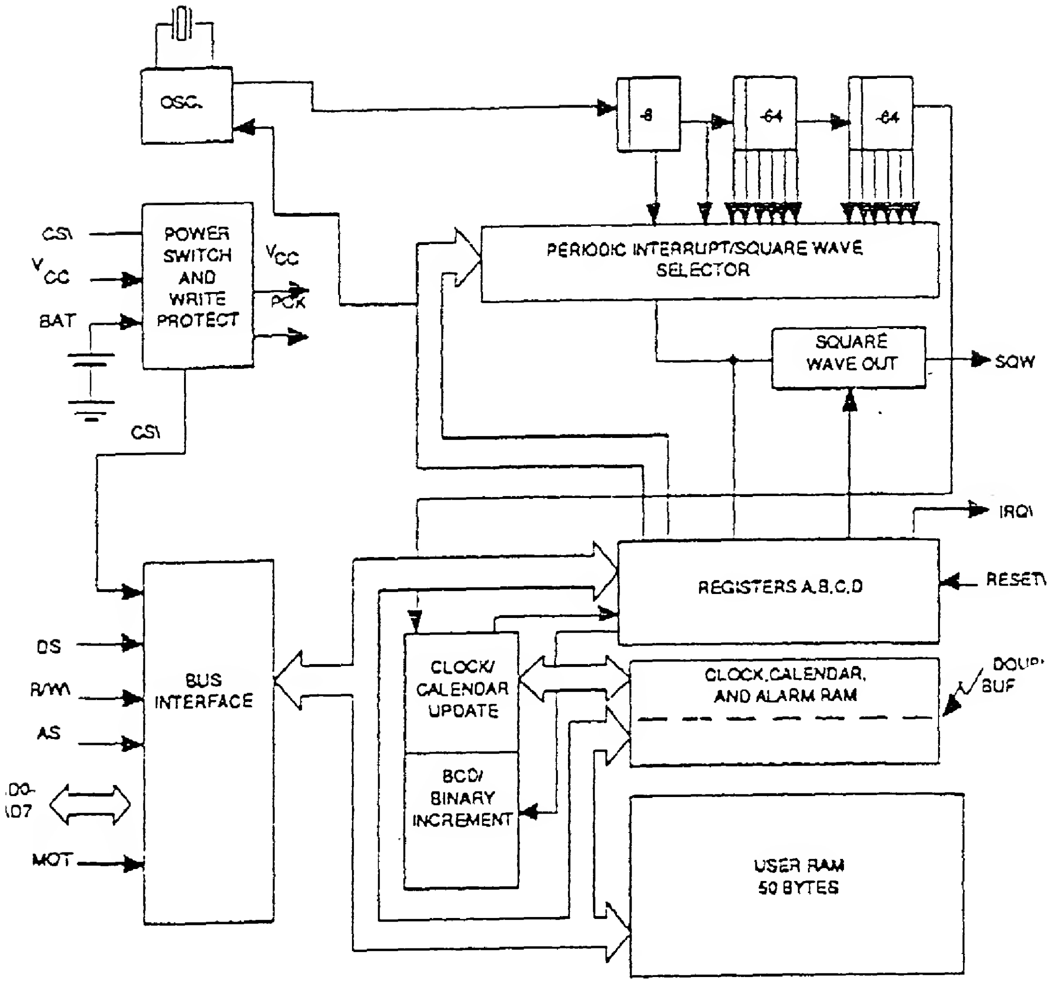


Figure 15 Real-Time Clock Block Diagram

conflicts if more than one application tries to store parameters here. Because of this, it is recommended that application or utility writers DO NOT use this RAM and consider it reserved for usage by the MM/1 hardware or operating system software.

Offset	Function	Decimal Range	Binary Mode	BCD Mode
0	Seconds	0 - 59	00 - 3B	00 - 59
1	Seconds/Alarm	0 - 59	00 - 3B	00 - 59
2	Minutes	0 - 59	00 - 3B	00 - 59
3	Minutes/Alarm	0 - 59	00 - 3B	00 - 59
4	Hours 24 hour mode	1 - 12 0 - 23	00 - 0C AM 81 - 8C PM 00 - 17	01 - 12 AM 81 - 92 PM 00 - 17
5	Hours/Alarm 24 hour mode	1 - 12 0 - 23	00 - 0C AM 81 - 8C PM 00 - 17	01 - 12 AM 81 - 92 PM 00 - 17
6	Day of Week	1 - 7	1 - 7	01 - 07
7	Date of Month	1 - 31	1 - 1F	01 - 31
8	Month	1 - 12	1 - C	01 - 12
9	Year	0 - 99	00 - 63	01 - 99

Table 20 Real Time Clock Offset Values

10.6 ADC/DAC Sound I/O

The MM/1 has the ability to process sound through a pair of 8-bit ADC/DAC chips. The Analog Devices AD7569 is a complete digital I/O system on a chip. It features a 2 microsecond analog to digital conversion time, a track/hold system with 200khz bandwidth, and automatic temperature compensation.

Interfacing to applications programs is easy. The sound I/O system occupies two bytes in the memory map at _$E00100 and _$E00101. Sending sound out is simply a matter of making sure the MC68230 general purpose control port has the Sound/Joystick bit set for sound I/O, and writing to the sound I/O device memory address. Note that the two adjacent addresses are for the two sound channels--address _$E00100 for channel A and address _$E00101 for channel B. An 8-bit, 2s

complement, sound word can be sent to each address independently, which is how stereo sound is achieved.

Reading (or recording) from the ADC is just as easy. However, all of the details for reading and writing these devices are taken care of in the MM/1 windowing system using I_$SS__Play and I_$SS__Record system calls. These calls are fully documented in the windowing system manuals.

The maximum line voltage levels for the ADC/DAC devices Input and Output is + or - 2.5 volts. The loudest playable sound files will generate a 2.5 Volt P-P sine wave. During sound record, this level will generate the binary code for the loudest volume levels.

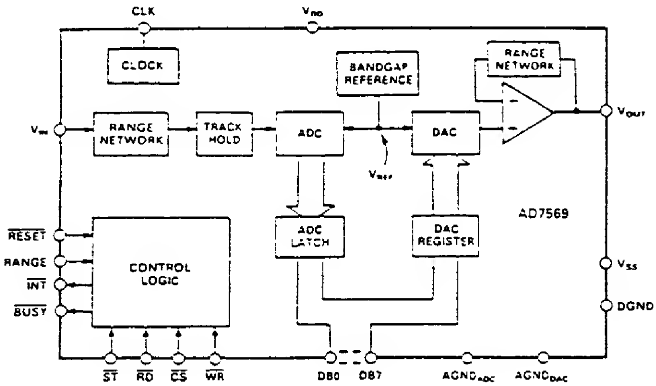


Figure 16 AD7569 Functional Block Diagram

End of Section

11 Keyboard Character Codes

The following chart gives the HEX code returned for each keyboard character on the MM/1. Note that the codes for control and alternate characters is given for those key combinations that return a code. Any left blank mean no code is returned for that key combination, or it is the same code as in the column to the left.

Char	Norm	Shift	Cntl	Alt	Char	Norm	Shift	Cntl	Alt
!	21			A1	U	55		15	D5
•	22			A2	V	56		16	D6
●	23			A3	W	57		17	D7
●	24			A4	X	5B		18	D9
●	25			A5	Y	59		15	D9
&	26			A6	Z	5A		1A	DA
'	27			A7	[	5B			DB
(	28			A8	\	5C	7C	1C	DC
)	29			A9	]	5D			DD
•	2A			AA	\	5E			DE
+	2B			AB	_	5F			DF
,	2C			AC	a	61	41	01	E1
-	2D	5F	1F	AD	b	62	42	02	E2
.	2E			AE	c	63	43	03	E3
/	2F			AF	X	64	44	04	E4
0	30	29		B0	e	65	45	05	E5
1	31	21		B1	f	66	46	06	E5
2	32	40		B2	g	67	47	07	E7

Char	Norm	Shift	Cntl	Alt	Char	Norm	Shift	Cntl	Alt
3	33	23		B3	h	68	4B	08	E8
4	34	24		B4	i	69	49	09	F0
B	35	25		B5	j	6A	4A	0A	FA
6	36	5E	1E	B6	k	6E	4B	08	EB
7	37	26		B7	r	6E	4C	0C	EC
8	33	2A		B8	m	6D	4D	0D	ED
9	39	26		B9	p	6E	4B	0F	EE
:	3A			0A	o	6E	4F	0F	EF
:	3B			BB	p	70	50	14	F0
<	3C			BC	u	71	59	11	F1
=	3D	2B		BD	r	72	52	12	F2
6	3B			BE	k	73	53	13	F3
:	3F			0F	r	74	50	14	F4
@	40			C0	u	75	55	15	F5
=	41		01	C1	h	76	56	14	F8
B	42		02	C2	w	77	57	17	F7
G	43		03	C5	p	78	56	14	F8
D	44		04	C4	y	78	59	19	F9
E	45		05	C5	z	7A	5A	1A	FA
F	46		09	C6	{	7B			FB
G	47		07	C7	|	7C			FC
H	48		08	C4	}	7D			FD
I	47		09	C5	r	7E			FE
I	4A		0A	CA	Esc	1B		1B	9B
K	4B		0B	CB	Home	01			81

Char	Norm	Shift	Cntl	Alt	Char	Norm	Shift	Cntl	Alt
L	4C		0C	CC	End	05			85
M	4D		0D	CD	Up Arrow	10			90
P	4E		0E	CE	Down Arrow	0E			8E
O	4F		0F	CF	Right Arrow	06			86
P	50		10	D0	Left Arrow	02			82
Q	51		11	D1	PgDn	16			96
R	52		12	D2	PgUp	1A			9A
S	53		13	D3	Ins	03			83
T	54		14	D4	Del	04			84

Keyboard Character Codes

11.1 Terminal Control Code Table

Many terminal functions such as moving the cursor, turning on and off reverse video, etc., have been traditionally controlled by sending certain control codes to the terminal to achieve the function desired.

In the case of all of the following codes, simply writing them to the current screen will cause that function to be performed.

Character	Code	Meaning
Null	0C	Do nothing - used for padding
Home	01	Home the cursor
GotoXY	02	Go to position XY (02 XX YY)
ErasLine	0C	Erase to end-of-line
ErasEOL	04	Erase to end-of-line (2nd method)
CurRght	0C	Cursor right
Bell	07	Sounds the bell
CurLeft	02	Cursor left
CurUp	09	Cursor up
CurDwn	0A	Cuursor down
ErasEOS	0B	Erase to end-of-screen
Cls	0C	Clear the entire screen
Crtn	0C	Carriage return
CurOff	05 20	Turn off the text cursor
CurOn	05 21	Turn the cursor back on
RevOn	1F 20	Turn on reverse video
RevOff	1F 21	Turn off reverse video
UndLnOn	1F 22	Turn underlining on
UndLnOff	1F 23	Turn underlining off
InsLine	1F 30	Insert a line at the cursor
DelLine	1F 31	Delete a line at the cursor
VT100 SavCur	1B 37	Save the cursor position
VT100 ResCur	1B 38	Restore the cursor position
VT100 Esc	1B 5B	esc [VT100 ESC sequence

Table 22 Cursor Control Codes

End of Section

12 Jumper Settings

12.1 Processor Board Jumper Settings

Jumper	Settings	Action
P2	None	DB-9 serial port /t0 See DB-9 serial port diagram for pin-out
P3	None	Power/Reset connector
P5	None	Not a jumper, but a sound output location. Pin 1 is ground, the rest are all sound output. Same as sound output on the CM-8 monitor.
P6	None	Alternate keyboard connector Pin 1 - Keyboard Clock Pin 2 - Keyboard data Pin 3 - RESET* Pin 4 - Ground Pin 5 - +5 Volts
P7		Memory jumpers

```
1 *  * 2
  |  |   Set for 3 Megabyte
3 *  * 4

1 *  * 2
         Set for 1 Megabyte
3 *  * 4
```

See the board layout diagram for pin locations

P8	OPEN	Sets logic of VSYNC* and HSYNC* to normally high or low. Depends upon monitor. Pins 1 and 2 - VSYNC* Pins 3 and 4 - HSYNC*
P9	None	Serial port /t1 Pin 1 - Transmit data Pin 2 - Receive data Pin 5 - +5 volts Pin 6 - Ground
P12	1<>2 ON 3<>4 OFF 5<>6 OFF	1 and 2 toggle the ROM monitor 3 and 4 set CTS* on /t1 5 and 6 set Master/Slave mode on the Inter-IC bus.
P13		Sets pin 2 of floppy drive to high or low depending on if the type of floppy drive needs it set
P14	None	Floppy disk connector
P15	None	Inter-IC connector Pin 1 - SDA (Serial data) Pin 2 - SCL (Serial clock)

12.2 I/O Board Jumper Settings

Jumper	Settings	Action
P1	None	Parallel port /p Pin 1 - STROBE* Pin 2 - BUSY Pin 3 - D0

		Pin 5 - D1 Pin 7 - D2 Pin 9 - D3 Pin 11 - D4 Pin 13 - D5 Pin 15 - D6 Pin 17 - D7 Pin 19 - ACK* Even numbered pins from 10 - 20 are grounded
P2	None	Serial port /t2
P5	1< >2 for /p 3< >4 for /p1	Toggles linefeed on or off
P6	None	Sound I/O connector Pin 1 - Chan B Sound In Pin 2 - Ground Pin 3 - Channel A Sound In Pin 4 - Channel B Sound Out Pin 5 - Channel A Sound Out See the pin-out appendix for a diagram
P7	None	SCSI activity light
P8	None	Joystick Connector - See the pin-out appendix for a diagram
P9	None	Serial port /t3 Pin 1 - Transmit Data Pin 2 - Receive Data Pin 3 - RTS* Pin 4 - CTS* Pin 5 - +5 volts Pin 6 - Ground Pin 7 - DCD

		Pin 8 - DTR
P10	None	Parallel port /p1 See P1 above for a pin-out
P11	3< >4	Adjusts refresh rate
P12	None	Serial port /t4 See P9 above for a pin-out
J1	1< >2	Selects SCSI power from the bus or from the I/O board. Only used if another SCSI MASTER is on the bus.
H1/H2	H1[4]< >H2[4]	Selects I/O wait states

12.3 Paddle Board Jumper Settings

Two connectors are provided on the board. P5 is a 6 pin single-inline connector that is designed to mate with the processors boards /t1 serial port. When using this port, only the send and receive data lines are brought out to the paddle board, along with +5 volts and ground. Connector P1 is an 8 position dual-header connector designed for use with the two MC68681 ports on the I/O board /t3 and /t4. When used in this configuration, all the modem control lines and well as send and receive data are brought out.

Jumper P2 controls the RS-232C function of the paddle board, whether it is a DTE (data terminal equipment) or a DCE (data communications equipment). Setting all the jumpers on P2 in a horizontal manner configures the port as DTE. Setting the jumpers vertically configures the port as DCE. See Table 22.

Jumper P4 controls the settings for RTS and DTR. One setting allows these control lines to toggle freely while the other setting sets either or both lines to a logical high. See Table 22.

The pin-out of the DB-9 serial connector is shown in the figure below with both DTE and DCE signals shown.

```
DTE              DCE

1 *--* 2         1 *  * 2
                   |  |
3 *--* 4         3 *  * 4

5 *--* 6         5 *  * 6
                   |  |
7 *--* 8         7 *  * 8

9 *--* 10        9 *  * 10
                   |  |
11 *--* 12       11 *  * 12
```

Figure 17 DTE/DCE Jumper Settings

```
RTS/DTR Toggled     RTS/DTR Tied High

1 *--* 2            1 *  * 2

3 *  * 4            3 *--* 4

5 *--* 6            5 *  * 6

7 *  * 8            7 *--* 8
```

Figure 18 RTS/DTR Jumper Settings

Pin 1 - Transmit Data
Pin 2 - Receive Data
Pin 5 - +5 volts
Pin 6 - Ground

Figure 19 6 Position Connector Pin-out

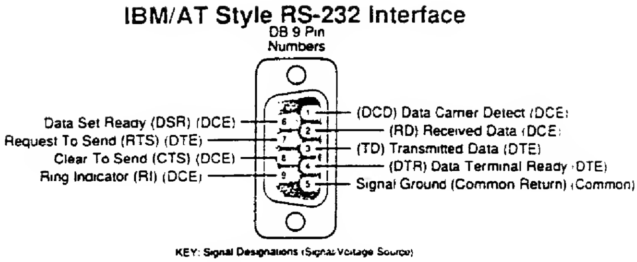


Figure 20 RS-232C Pin-outs

End of Section

13.1 Video Port

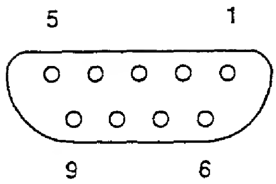


Figure 21 Video Port Diagram

Pin	Function
1	Ground
2	Ground
3	Red
3	Green
5	Blue
6	N/C
7	Sound
6	HSync
9	VSync

Table 23 Video Port Pin Specifications

13.2 Parallel Port

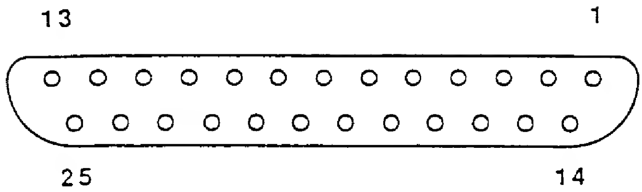


Figure 22 Parallel Port Diagram

13.3 Joystick Controller

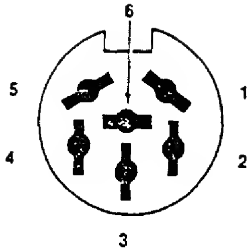


Figure 23 Joystick Controller Diagram

Pin	Function
1	Right-Left Input
2	Up-Down Input
3	Ground
4	Fire Button #1
5	+5 volts
6	Fire Button #2

Table 24 Joystick Pin Specifications

13.4 Keyboard

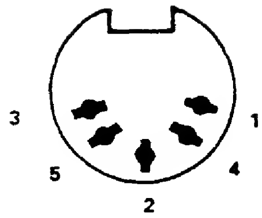


Figure 24 Keyboard Socket Diagram

Pin	Function
1	Clock
2	Data
3	Reset
4	Ground
5	+5 volts

Table 25 Keyboard Port Pin Specifications

13.5 IBM PC Color Monitor Interface

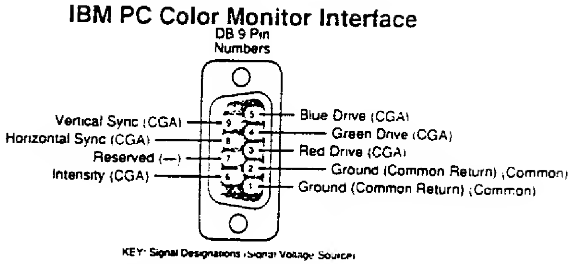


Figure 25 IBM PC Color Monitor Interface

13.6 EIA-232 Interface Reference

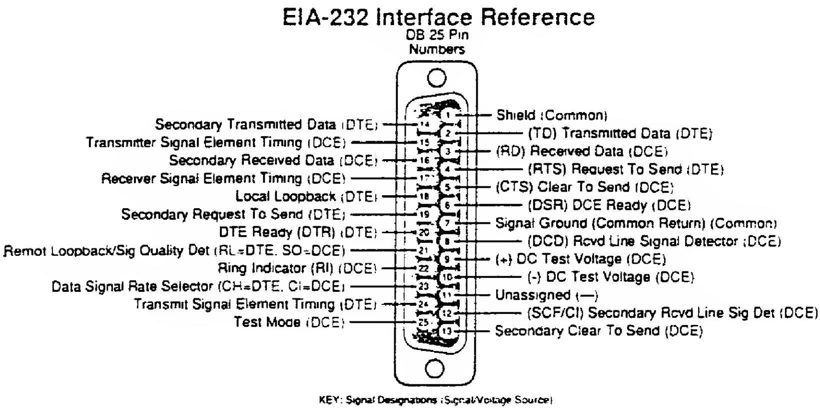


Figure 26 EIA-232 Interface Reference

13.7 Parallel Interface

Centronics Parallel Interface Reference

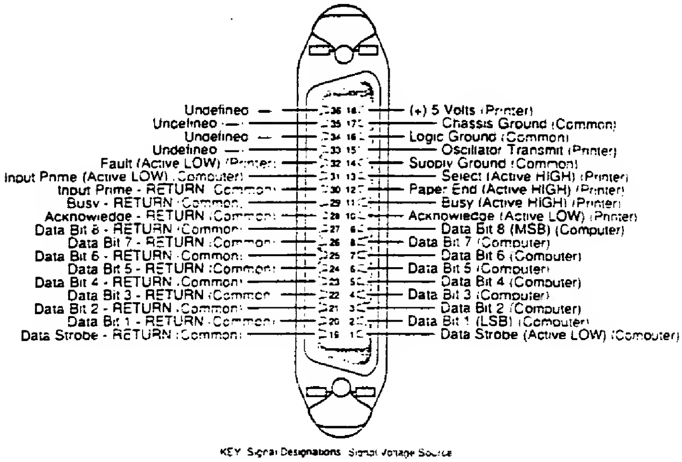


Figure 27 Centronics Parallel Interface Reference

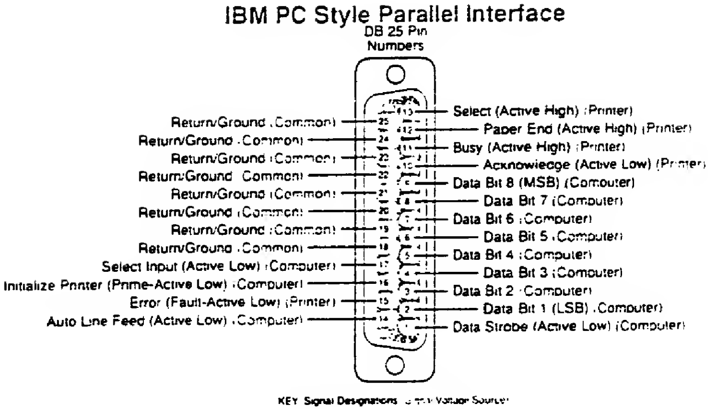


Figure 28 IBM PC Style Parallel Interface

13.8 Differences Between DCD/DTE

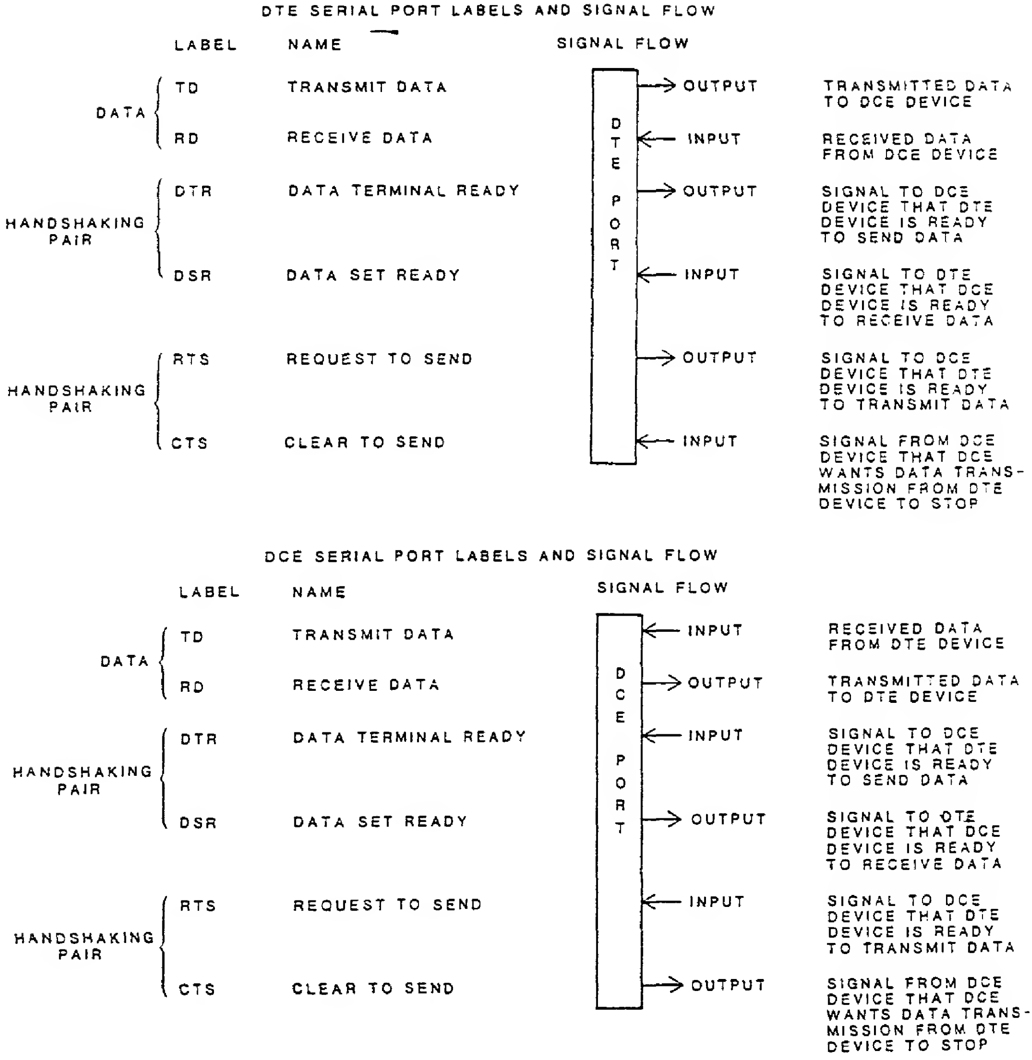


Figure 29 Differences Between DCD/DTE

13.9 DTE/DCD Handshaking

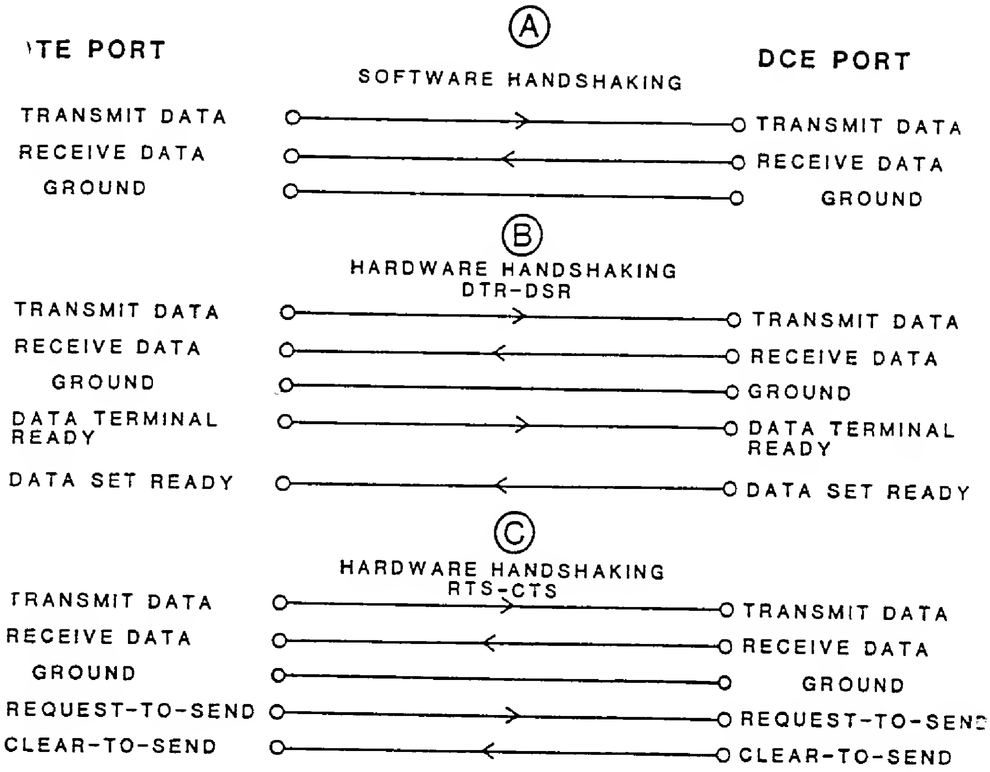


Figure 30 DTE/DCD Handshaking

13.10 Parallel Signals

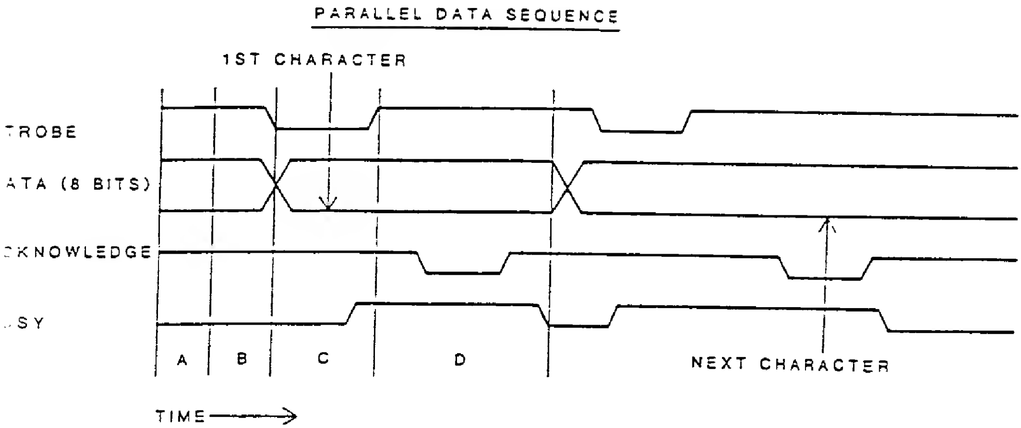


Figure 31 Parallel Data Sequence

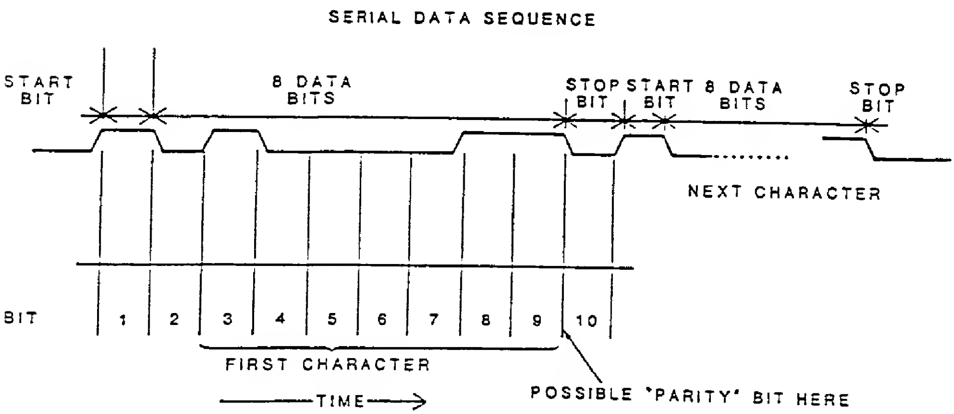


Figure 33 Serial Data Sequence

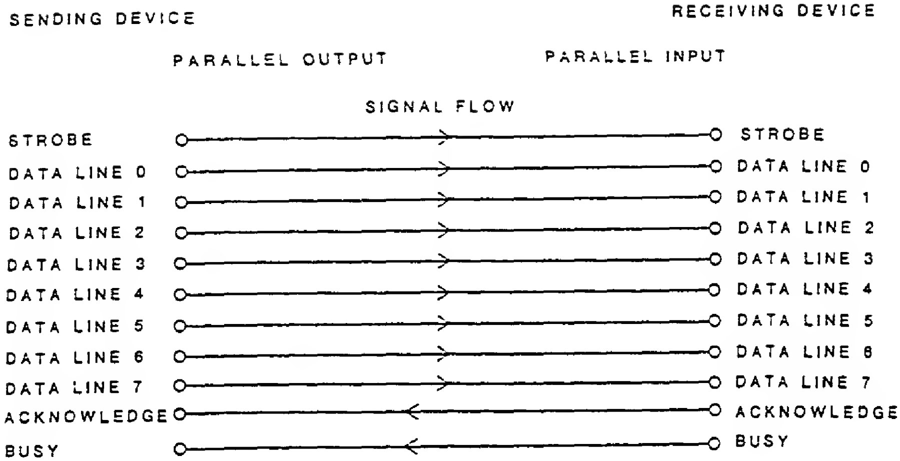


Figure 32 Parallel Signal Flow

Power CONNECTOR

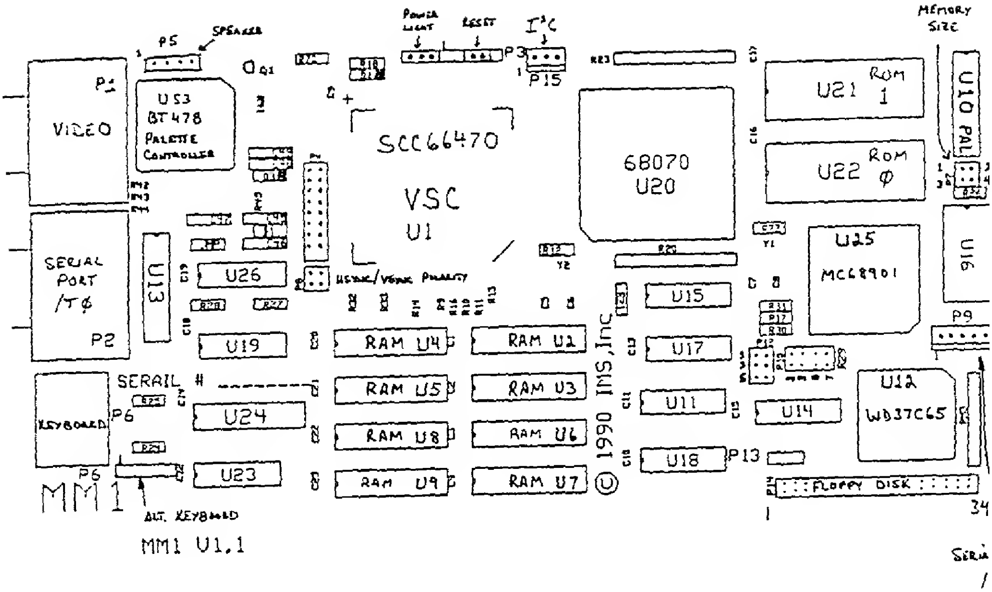


Figure 34 MM/1 Processor Board Layout

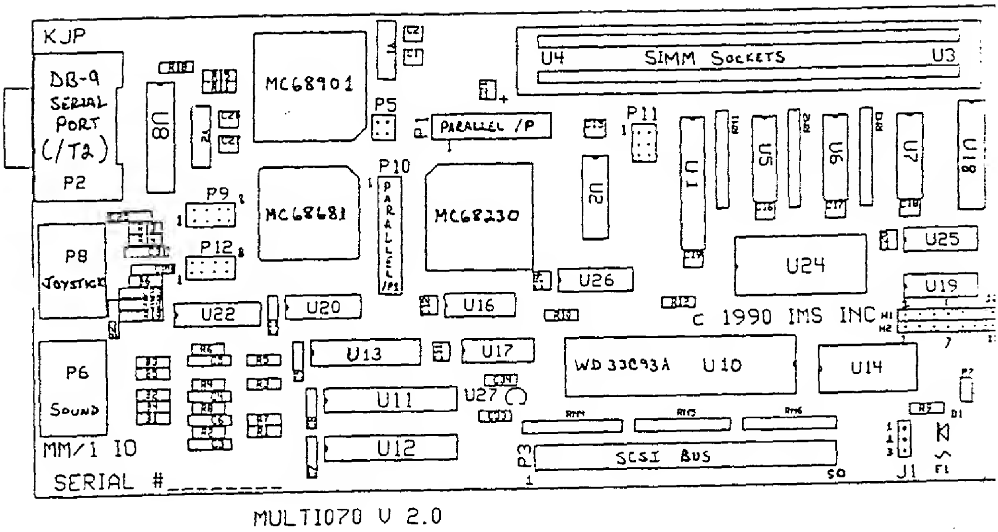


Figure 35 MM/1 I/O Board Layout

End of Section

14 References

Motorola Document MC68901/D "Multi-Function Peripheral" 1988

Motorola Document MC68681 "Dual Asynchronous Receiver/Transmitter (DUART)", 1985

Motorola Document MC68230 "Parallel Interface/Timer (PI/T)", 1983

Motorola Document MC145407/D "5-Volt-Only Driver/Receiver", 1989

Philips Components Product Specification "SCC68070 16/32-bit microprocessor", November 1990

Philips Components SCC68070 User Manual Part 1 - Hardware
Philips Components SCC68070 User Manual Part 2 - Software

Philips Components Product Specification "SCC66470 Video and System Controller", December 1989

Philips/Signetics Application Notes for the SCC68070, August 1990

Philips Components SCC66470 User Manual, 1990

Philips/Signetics Application Notes for the SCC66470, Jun 1990

Dallas Semiconductors Data Sheet "DS1287 Real Time Clock"

Analog Devices Data Sheet "AD7569/AD7669 Complete, 8-Bit Analog I/O System"

Western Digital Data Sheet "WD37C65/A/B Floppy Disk Subsystem Controller", 1988

Application Notes for the WD37C65/A Floppy Disk Subsystem Controller

Western Digital Data Sheet "WD33C92 and WD33C93 SCSI-Bus Interface Controller", 1987

Western Digital Advance Information "WD33C93B Enhanced SCSI Bus Interface Controller", December 1990

End of Section

15 Manufacturer Addresses

Analog Devices
1 Technology Way
P.O. Box 9106
Norwood MA 02062-9106
(617) 329-4700

Brooktree Corp.
9950 Barnes Canyon Road
San Diego CA 92121
(619) 452-8270
(800) 843-3642

Motorola Literature Distribution
P.O. Box 20912
Phoenix AZ 85036

Philips Components
Discrete Products Division
2001 West Blue Heron Blvd.
P.O. Box 10330
Riviera Beach FL 33404
(407) 881-3200

Signetics Corp.
811 East Arques Ave.
Sunnyvale CA 94088-3409
(408) 991-2000

Western Digital
2445 McCabe Way
Irvine CA 92714
(800) 847-6181